8-12-1983

Interactive computer program

Bruce Howard Browne

ROCHESTER INSTITUTE OF TECHNOLOGY

A Thesis Submitted to the Faculty of
The College of Fine and Applied Arts
in Candidacy for the Degree of
MASTER OF FINE ARTS

Interactive Computer Program

by Bruce Howard Browne

August 12, 1983

Approvals

Chief Advisor: James C. Ver Hague

Date: August 9, 1983

Associate Advisor: R. Roger Remington

Date: August 9, 1983

Associate Advisor: Evelyn Culbertson

Date: Aug 8, 1983

Graduate Academic Council Representative:

Fred Meyer

Date: 9/12/83

Dean, College of Fine and Applied Arts:

Dr. Robert H. Johnston

Date: 9/12/83

I Bruce Howard Browne, prefer to be contacted each time a request for production is made. I can be reached at the following address:

Date: Aug 8, 1983

Bruce H. Browne

THESIS COMMITTEE

James C. Ver Hague, Chief Advisor

R. Roger Remington

Evelyn Culbertson

ACKNOWLEDGEMENTS

I would like to thank Gary Roshak, who helped me to achieve more than either one of us could have accomplished individually. Special thanks should go to Jim Ver Hague for all his help and encouragement and to Evie Culbertson for introducing me to Gary.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF APPENDICES

INTRODUCTION

The purpose of this thesis was to design an interactive computer program to introduce designers to the new physical and conceptual space that computer technology presents. It was also intended as an example of graphic design principles applied to the design of electronic information and to promote computer literacy in the design field. The program was meant to be an introduction to computers and their relationship to the field of graphic design.

The thesis topic grew out of my research into computer graphics and my belief that designers have a social responsibility to help shape the environment. I realized that computer technology was playing an increasingly important role in our lives but that graphic designers were doing very little to influence

the shape or use of these new technologies.

I began research for a thesis topic in June of 1982. I was initially intrigued with the flashy animation that was being produced with state-of-the-art computer graphics systems. I had been exposed to many concepts on computers at The Designer and the Technology Explosion conference held at RIT and through many books and articles which I had read on the subject. All of the information to which I had been exposed was fermenting in my mind and it was eventually synthesized into my own understanding of computers.

I started to see something beneath the surface of computer graphics imagery that was much more important than the slick images being created. I posed several questions to myself which included: what implications does the information revolution have on the field of graphic design and society in general; how do design and computer technology relate to each other and where are the points of synthesis; how can computers be applied to education; what implications does the increased speed of communication have on the design process; how can computers and design be integrated to better serve

each other; what social responsibility does the information revolution place on designers; and what does the time/space relationship of computer information have on graphic design? It appeared that computers would have a profound impact on all of these questions and that designers should be involved in their outcome.

From these questions I developed the idea that the way in which people interacted with computers was more important than what was being created with them. I also saw that the visual display of information on computer systems used for everyday storage and retrieval of information, such as office computers, was being ignored. This suggested that there was a great need for information graphics in the emerging computer technology. I also recognized that the interactivity of computers offered great potential for using computers as educational tools.

The idea of an interactive computer program emerged from these concepts. I wanted the thesis project to be an environment for learning and creating more than an end product in itself. Simultaneously, I intended for it to be an example of graphic design principles applied to computer

systems. Therefore I had to use the computer and hence an interactive program.

BACKGROUND

The graphic design field is faced with revolutionary changes in both the tools it uses and the products it designs. Computer technology is changing our society from an industrial one to an information-based one. This shift in technology has many implications for the design field. The quantity and complexity of information is growing at a tremendous rate which magnifies the need for effective communication. Information will be stored and disseminated electronically through computer systems which will require the design of vast quantities of information. Designers will be designing information that may never appear in printed form but exist solely as electronic information. They will also have to apply their

expertise to the design of computer systems and the way in which people communicate with them. It is not enough just to have information; there must be an easy way to find and use it. Information graphics will be the greatest challenge for designers in the computer age.

As the complexity of information that the designer must communicate increases, faster, better tools will become a necessity. Computer graphics systems can be very powerful design tools because of their speed and flexibility. It is possible to manipulate images and generate alternative designs very fast. Computers can free the creative person from the drudgery of drawing out ideas. It is possible to create more sketches in a shorter amount of time which will undoubtedly lead to better solutions and increased productivity. Computers also have the potential of giving designers control over the complete process of producing products and of allowing them to see exactly what the finished pieces will look like.

It is important to understand that computers only carry out instructions from users. Computers cannot differentiate between good and bad design,

yet. Computers are simply tools that designers can use like a t-square or triangle only much more powerful. Design is design whether you use a pencil or an electronic stylus.

Computers may require that the designer have a better understanding of the systematic approach to creative problem-solving, which combines logic and intuition. This will be necessary because the designer will be faced with many decisions more frequently and it will be difficult to make these decisions without an orderly thought process.

Although computer graphics systems have great potential, most systems currently lack essential features for layout work and communication is often awkward. This situation is changing and will change even more as designers become involved in the new technologies.

THE THREE FACES

Designer and computer graphics consultant Aaron Marcus has developed a concept that computer systems can be divided into three components or faces. The three faces are the Outerface, Interface, and Innerface. Each face is an area in which designers can become involved to improve the communication of electronic information.

I chose the general conceptual framework of Marcus' three faces because it was a simple, clear method of introducing designers to computers. It was very useful to break a complex tool like the computer into smaller understandable segments. The three faces also provided concrete examples of where graphic design principles could be applied to computer technology.

As I was researching Marcus' concepts, I discovered that there was a degree of ambiguity in the breakdown and definition of the three faces. For example, each face actually contains components of the other faces and could be broken down further. The three faces appeared to be an arbitrary division of the computer but I still believed that it was a sound concept. I also thought that it was better to use an existing concept rather than to create my own arbitrary one. This was done for the sake of standardization which is greatly lacking in the computer graphics field.

Although the three faces were a useful concept for introducing computers and graphic design, it was only a beginning. I believed that design should go beyond the superficial aspects of the computer and deal with the computer as a whole to understand the technical problems which limit their graphic capabilities. Once designers grasp these problems they can work to solve them and create new tools which are better suited to the graphic designer and society as a whole.

Outerface

An Outerface is the output or end product of communication with the computer. In computer graphic systems the outerface is an image created through interfaces. All computers are in a sense computer graphics systems because they display information in some form.

Computers can generate many types of images including charts, graphs, text, maps, photos, symbols, animation, and illustration. Images can be displayed on a screen, either the computer's monitor or television screen, on paper from printers and plotters, or on film for slides and animation. Outerfaces are traditional design problems such as page layouts and graphs that can be solved very effectively on the computer.

The resolution (fineness or coarseness of the dot pattern) of the computer monitor and of the hardcopy output device are important considerations when designing Outerfaces. They have an effect on the sophistication of images that can be used. Low resolution monitors require simple shapes and large type because the image quality is poor. Curvilinear

forms appear jagged and it is difficult to define small objects. High resolution monitors can display complex organic and geometric forms with better image quality. Different computer systems have varying color capabilities from one to sixteen million depending on the sophistication of the system.

Printers generally have relatively low resolution output while plotters can render high resolution line drawings. Many have limited color capabilities depending on the type of system. Capturing the image on film is a common way of obtaining high resolution output. This can be done by shooting directly from the monitor with a camera or through the use of a film recording device. Images can also be stored on a magnetic disk or tape for later reproduction.

Resolution is also important when choosing type size and style. Typefaces often look awkward because the image quality of the monitor is not high enough for small type or finely detailed letterforms. Currently there are few typefaces designed specifically for computers. This is an area in which designers can make contributions.

Designers also have another important task with

Outerfaces. They can provide standards and formats that can be built into software which enable computers to design images automatically or semi-automatically. In the future there will be so much information that it will be impossible for designers to design it all. It will be necessary to transfer some of the designer's expertise to the computer.

Interface

An interface is a device or structure for communication between a user and computer for the creation of images. It appears on the monitor as a list of options, called a menu, or as a series of questions and answers. Interfaces allow the user to select and manipulate images and information. Depending on the type of computer system, selection can be made through a keyboard, electronic pen, puck, joystick, touch-sensitive screen, or voice input.

Interfaces are perhaps the most important aspect of a computer because they stand between the user and the power of a computer. The speed and

versatility of a computer are useless if communication with them is difficult.

Designers must become involved with Interfaces to make them more natural and easier to use. They can do this by improving the visual and mechanical communication devices such as menus, tablet overlays, and keyboard overlays. There has been a lack of design principles applied to information displayed on computer monitors. This is also true of the documentation and instructions that accompany these systems. The traditional design considerations that are used in composing a printed page should be applied to information on the monitor.

Designers can also create the conceptual structures of information. Many computer programs are essentially non-linear books which let the user jump around in information at different levels. These programs often have tree-like structures that branch out into different modules of information. Graphic design can visually reinforce these structures by making them more apparent so that it is easier to move through information. There are three important pieces of information that should appear in each image on the Monitor. As a user, you must be

able to discern where you are in the information structure, where you have been, and what options are available next. These visual clues will give the user a sense of orientation while moving through complex information.

Innerface

An Innerface is the visual display of internal instructions, or programs, that enables a computer to carry out tasks. Electronic circuitry (hardware) cannot function without these programming languages (software) that are translated into binary electronic pulses or bits of information. There are different levels of programming languages, from machine and assembly languages to high level programming languages. These languages communicate with the computer at different levels of abstraction.

Innerfaces are an important potential design area. The visual display of this information that controls the computer has been ignored until recently. Computers and software are becoming more complex, which necessitate simplifying the display of

instructions if programmers are to work efficiently.

Today it is often very difficult for a programmer to quickly understand the function of a program at a glance. The programs that run the computers of large corporations and government agencies are very complex. When the programmers who write these programs are no longer there it becomes a difficult task to decipher them.

Graphic designers can make it easier for computer scientists to read, understand, and debug programs by improving their visual format. Through the use of typography, symbols, color, and visual structures, designers can improve the legibility and communicative potential of programs.

Programming languages are evolving towards more natural human language and eventually to pure symbolic languages. Graphic designers with an understanding of programming languages could collaborate with computer scientists to develop new programming languages that would be easier to use. This is possible because designers are concerned with the manipulation of symbols to communicate information. Although creating programming languages is a very complex task, graphic design is one of the

best qualified professions to impact on programming languages. Technical knowledge of computers will enable graphic designers to play an integral part in the development of simple powerful languages.

THE SYMBOLS

One quarter prior to the beginning of my thesis I started to develop symbols that would represent each of the three faces of computer systems. The time involved in generating symbols prompted me to have the symbols designed at the outset of the thesis quarter. I believed that this would enable me to simply plug the symbols into a format and allow more time for other aspects of the program.

The symbols were designed with a semiotic structure which is used to analyze the three properties of visual imagery. The three parts of semiotics are the semantic, syntactic, and pragmatic. I also formulated very specific objectives which related to the symbols' eventual use in an interactive program (fig. 1). These objectives

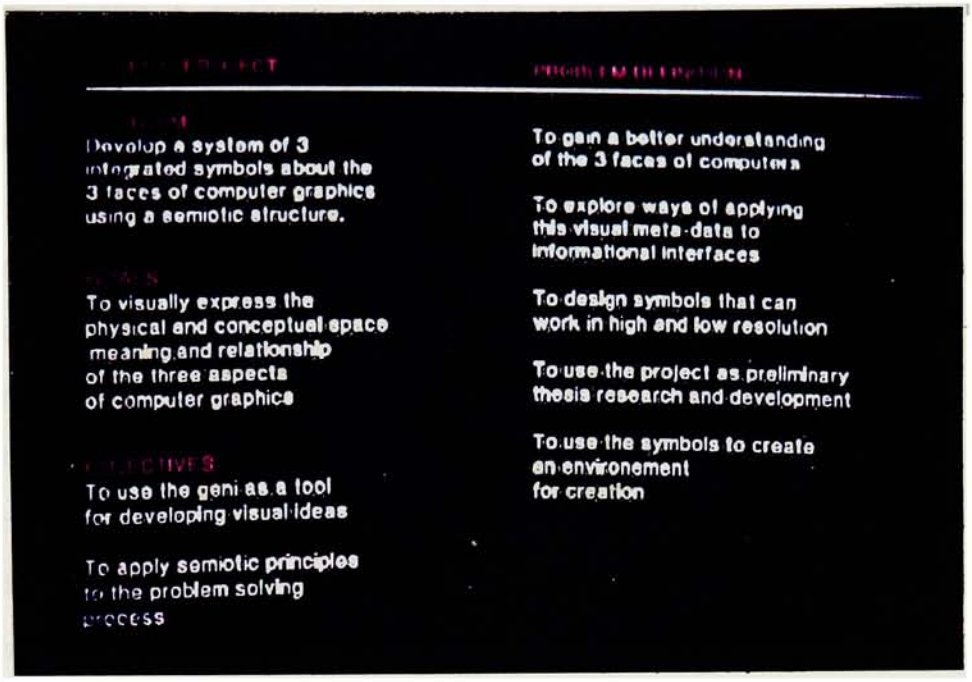


Fig. 1: Symbol Objectives

provided a criteria to judge the various approaches to solving the problem and ensured that the symbols would function in a menu environment.

I generated the symbols using the Genigraphics system in an attempt to demonstrate that the computer could be used as a tool for the entire design process. In reality this turned out to be difficult because of time restraints caused by heavy user demand. Nevertheless it did function well as a tool for the entire design process. The Genigraphics also served to check the image quality that would hold up on a medium resolution monitor. This helped to give me an idea of how simple or complex the symbols would have to be.

I spent a great deal of time researching the three faces, collecting images, making word lists, developing syntax charts, and creating conceptual maps. The next step was to apply visual form to the information which I had collected. My first attempts were based on abstract forms and concepts. The general concept was to visualize the invisible processes of the computer through a visual metamorphosis that would show the transition from the Inner to the Outer faces (see fig. 2). I applied

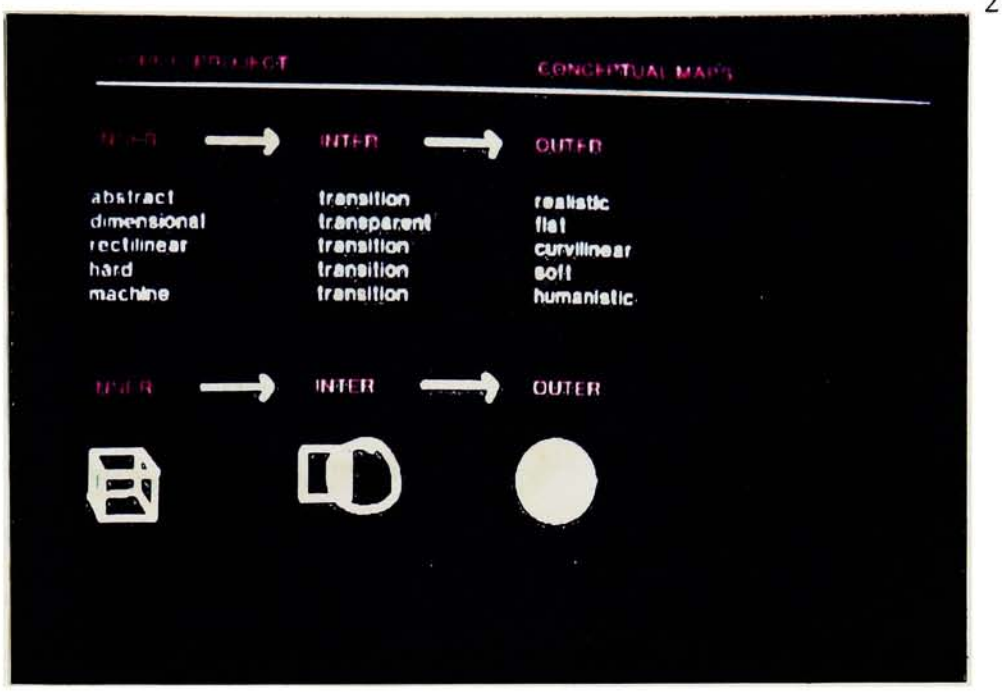


Fig. 2: Conceptual Map

visual forms from my syntax list to this concept and worked out three preliminary symbols.

I found that while I was designing some interesting visual forms, I was not communicating the meaning clearly with the symbols. Upon realizing this, I set out to re-evaluate my concepts and objectives.

The symbols had to function in a menu situation which meant that they had to be able to be easily understood because the user would be selecting information based on them. In a menu environment the user must be able to make selections quickly and easily. Another important factor in the re-evaluation was the target audience. The program was intended to be used by student and professional designers with little or no computer experience. This suggested that the audience could not be expected to relate unfamiliar concepts to abstract forms.

Based on this criteria, I decided that pictograms or the use of representational forms would be more appropriate for the function of the symbols. Therefore I tried to select an object or action that would communicate the essence of each face. The

general concept was that each face entailed human interaction but that it was at different levels. One of my objectives was to illustrate the conceptual level of each face, from the micro to the macro view.

I experimented with a close-up of a computer chip to demonstrate the Innerface of the computer, a human figure with a monitor for the Interface, and a human figure with a printout for the Outerface. At this point the symbols lacked a unifying element to make them work together as a group. I went back to my concept that the common link between each face was human interaction. I used the hand in relation to the essential element of each face to visualize this idea. I chose a hand holding a chip for Innerface, a hand with a finger touching a keyboard for Interface, and a hand pulling a printout for Outerface (see fig. 3).

I realized that these symbols did not necessarily embody all of the aspects of each face but I did not see a possibility of universal symbols. An attempt was made to select the most common images of each face that would communicate to the layperson. For example, there are many ways of interacting with computers other than through a keyboard and key

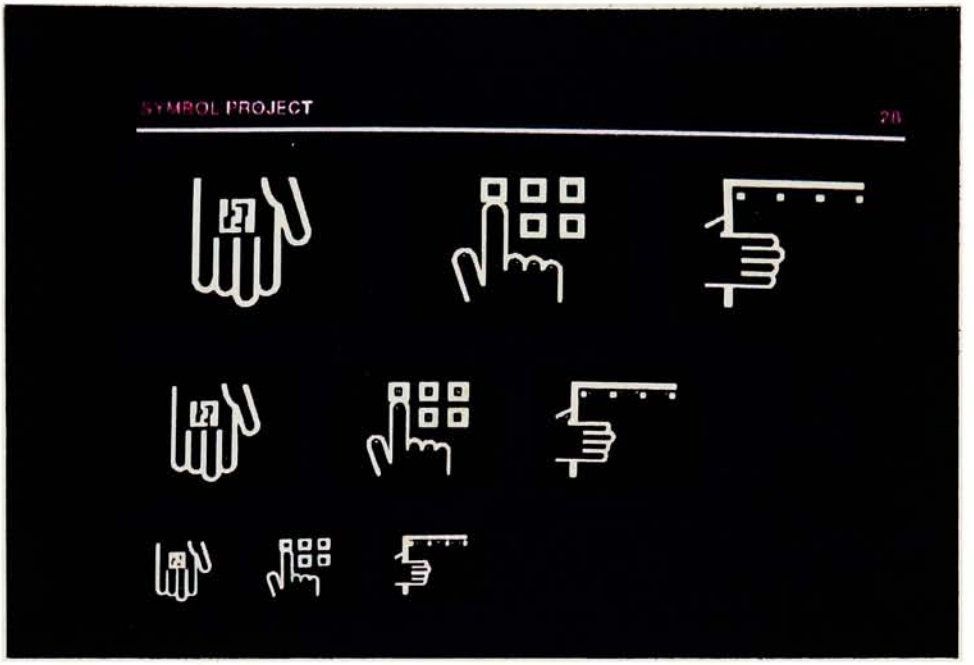


Fig. 3: Preliminary Symbols

boards may eventually become obsolete. But, the fact is that the keyboard is still more easily recognizable as an input device for the computer and holds more meaning than an electronic pen or puck. The symbol for Outerface does not explicitly indicate Outerfaces on monitors but it is implicit in that it represents the end products of interaction with the computer. I used a microprocessing chip to represent Innerfaces because I felt it would be understood as having something to do with the internal workings of the computer, just as the paper printout implied an end product or outward communication.

DEVELOPMENT OF THE PROGRAM

I began to develop the interactive program in December 1982. A first step was to decide specifically what information was to be included in the program and what functions it was to have. Another major decision was to choose the equipment that would be used to carry out the program. As I worked out the capabilities that I wanted the program to have it became obvious that I did not have the technical knowledge of programming or of computer systems to design an interactive program.

Fortunately, I was introduced to Gary Roshak, a student in the Department of Computer Science. Gary was familiar with Aaron Marcus' concept of the three faces of computer systems and he realized the importance of applying graphic design principles to

electronic information. I explained to Gary what I intended to do and he agreed to work with me on developing the program. He was responsible for the technical aspect of the program which included programming and hardware problems.

Hardware

We established several criteria for choosing the hardware that was to be used for the program. The ideal system needed to be programmable, to be high resolution, to have graphics capabilities, to be capable of character manipulation, to have a graphics tablet, and to have an electronic pen or puck. Next we discussed the systems that were available to us and which best fit the criteria. The systems that we explored using included: the Apple, Gigi, DEC 350 PC, Genigraphics 100C, and a Chromatics. The Genigraphics met all of the criteria except that it was not programmable. Gary attempted to obtain information on how images are stored and accessed on the Genigraphics. We believed that it would be possible to transfer the bit maps of single frames

from the Genigraphics to a DEC VAX 11/780 system and run the program on a personal computer. This idea proved to be impossible because Genigraphics would not divulge any information about their system.

Gigi and Apple were ruled out because of their low resolution, crude typographic characters, and limited color capability. The Apple was also not compatible with the VAX that Gary was using to write the program code.

The DEC PRO 350 personal computer appeared to be the best alternative to the Genigraphics. It had the highest resolution (960x240), it was compatible with the VAX, it had a graphics option with color capabilities, and the Computer Science Departent was obtaining several. The only drawback was that they were not yet available at RIT. It was not a crucial factor because we would not be ready for programming specific frames of information until I had completed the design of the program which I did not expect to finish until the end of winter quarter. During this time Gary was involved in finding specific information about the DEC 350 and in formulating his approach to solving the programming.

Text

The first step in writing the text for the program was to determine the information that was to be included. I made an outline of the information that I wished to present and proceeded to write text to fit the outline (see fig. 4).

The text basically broke down into four categories. There was an introduction and information about each of the three faces. Each of the general categories was further sub-divided into specific headings. I worked on refining this outline until I arrived at a working solution. The three faces each had a definition, information on designing for each face, examples of each face, and a glossary of terms relating to each face. The original outline also included a work space that was to allow the user to interactively create images. I later had to abandon this idea because it required too much complex programming and we lacked the proper hardware.

I decided to leave out as much technical language as possible because the program was meant to

PROGRAM OUTLINE

Introduction

- Explanation of program
  - purpose of program
  - outline or diagram of program
- Background of computers affect on Design Field
  - history and growth of computer industry
  - the information revolution and it's effect on society
  - the interface of graphic design and computers
- Instructions if needed
- Credits

Main Program

- Innerface
  - definition
  - 3 faces of innerface
  - verbal examples
  - visual examples
  - information on designing innerface
    - demonstrate where design skills are needed
    - examples and how they can be improved
    - provide work space to design innerface
- Interface
  - definition
  - 3 faces of interface
  - verbal examples
  - visual examples
  - information on designing an interface
    - demonstrate where design skills are needed
    - examples and how they can be improved
    - provide work space to design interface
- Outerface
  - definition
  - 3 faces of outerface
  - verbal examples
  - visual examples
  - information on designing an outerface
    - demonstrate where design skills are needed
    - examples and how they can be improved
    - provide work space to design outerface
- Glossary
  - computer terms

be an introduction. The text of the program was meant to communicate the basic concepts involved in a simple, clear method. I wrote in brief, concise segments to avoid putting too much text on any one frame of information and to provide more flexibility. A modular approach gave the possibility of adding and subtracting frames of information as well as changing their order. Less text on a single frame also allowed for larger point sizes of type. Larger type is more easily defined on a monitor because more pixels can be used to display its shape.

Structure

Determining the structure of the program was one of the most difficult tasks that I encountered. It required thinking about relationships through time and space. I found it to be one of the most profoundly different requirements of designing for computers.

The problem was that unlike printed material where the order is forced on the reader, computer information is accessible at any point. This

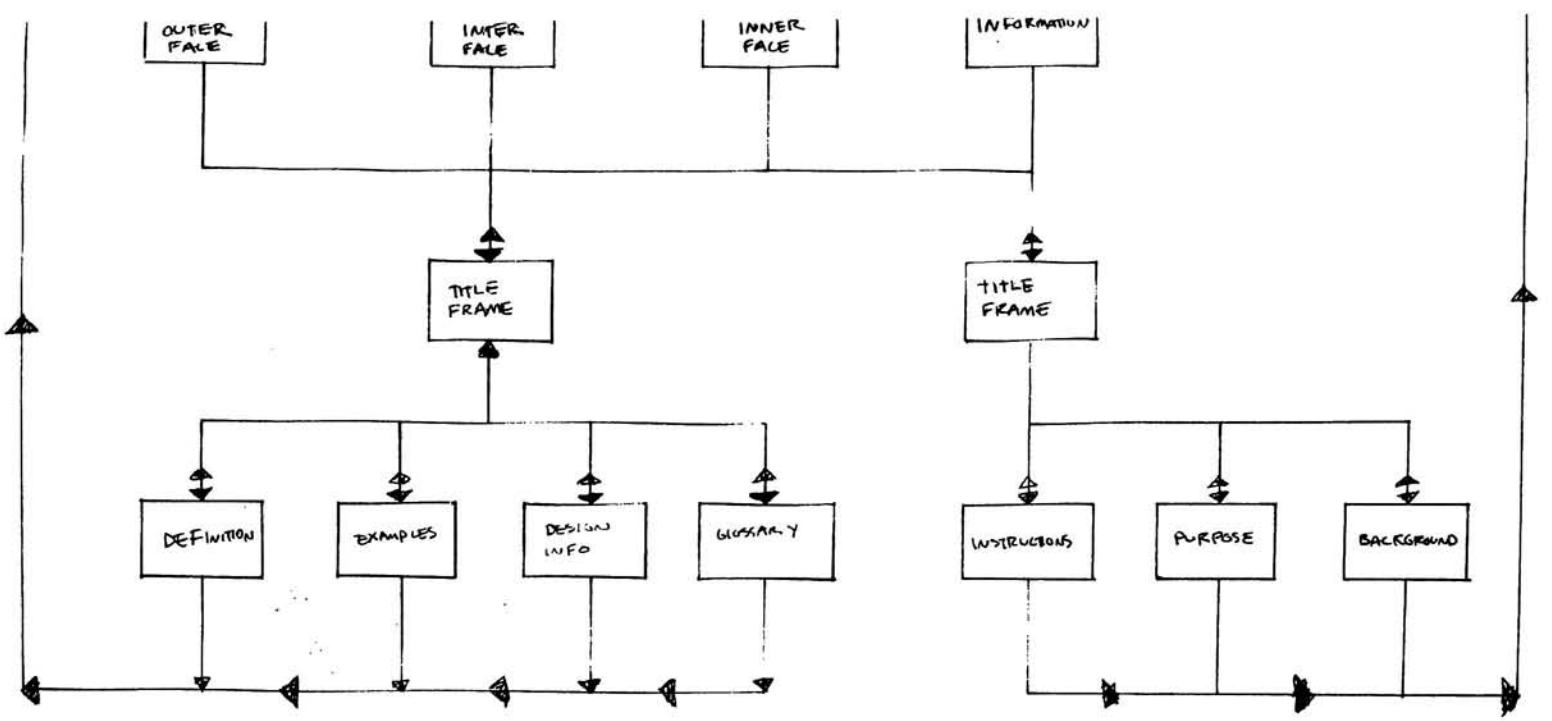
necessitates designing the structure in a coherent, understandable manner. The user should have maximum flexibility with minimum confusion.

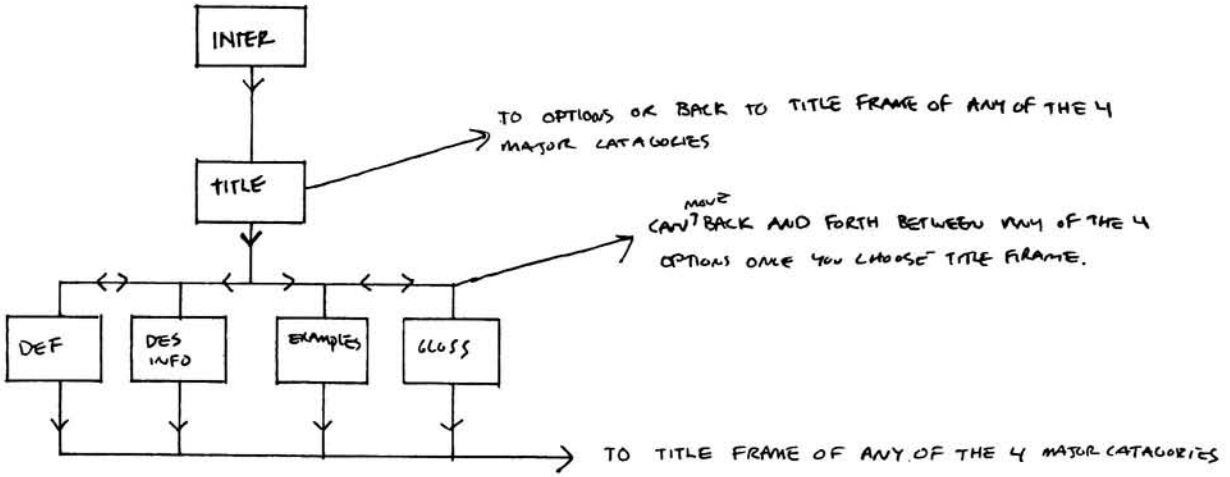
The outline of the text suggested the breakup of the information into four general categories with additional subcategories. I thought of the program as modules of information which were connected to each other. I explored many different ways of connecting these modules (see fig. 5). It became clear that I would have to place some restrictions on the order in which the user could view information.

Through the exploration of different structures I developed several parameters that the structure of the program should meet. Instructions should be the first item that comes on the monitor when the system is turned on or booted. The user who is already familiar with the program should have the option of moving out of the instructions to any other option. The user should also be able to choose any one of the four major categories from any frame. Once the user has selected one of the major categories it should be possible to choose any option and view it in sequence. The user should be able to move backward or forward one frame at a time in any option.

① CAN CHOOSE TITLE FRAME OF ANY OF THE 4 MAJOR CATAGORIES FROM ANY FRAME

② CAN CHOOSE ANY ONE OF THE 4 OPTIONS UNDER 3 FACES OR THE 3 OPTIONS UNDER INFORMATION FROM THE ~~FOUR~~ FRAMES UNDER THAT FACE AFTER CHOOSING TITLE FRAME OF THAT FACE.





① INFORMATION
② OUTERFACE
③ INTERFACE
④ INNERFACE

FORMAT

The development of the format was the point of synthesis in the thesis project because it brought together all the elements of the program. I drew many thumbnail sketches which combined the symbols, text, and structure into a format that would work for an interactive program. From the rough sketches I started full size rough layouts.

I used the Genigraphics to create the prototype layouts. At that point in time the Genigraphics was possibly going to be used for producing images which would be transfered to the VAX. I continued to use the Genigraphics to experiment with layouts even after it was established that it would be impossible to use for the final program. The Genigraphics turned out to be an excellent tool for doing layouts. I decided that it would be good to use the Genigraphics to develop images for an "ideal" program because the question of hardware was still up in the air. I worked on the Genigraphics as if we would be able to program it but I understood that my images

would have to be adapted to the equipment we used for the program.

First, I had to determine the exact size and proportions of the total image area that would be on the monitor. It was difficult to decide what size the single frames should be because I did not yet know on what monitor the images would finally be displayed. This was important because different computers display images in different proportions on monitors. Some of the monitors I examined bled images off the top or sides. I decided to treat the frames like pages of information and to mask off the unused portion of the monitor. This would enable the proportion to remain constant no matter what system it was displayed on and would contain the information in something resembling a page.

A 3:2 aspect ratio was used for the frame dimensions because those are the proportions of a 35 mm slide, which the Genigraphics was designed to produce. It is also an aesthetically pleasing proportion that takes advantage of the horizontal emphasis of the monitor while maximizing the use of space. Temporarily, the size was taken from a full screen regeneration of the Genigraphics which is

9.75x6.5 inches.

During the evolution of the format I was constantly switching between paper and monitor to change layouts. I dealt with this problem from the beginning because I did not have enough computer time to create layouts exclusively on the computer. This caused a problem because I had to have some way of translating layouts that I did on paper to the computer and vice versa. I also had to know what relationship Geni units had to point sizes because type on the Genigraphics is measured in Geni units.

Luckily, Genigraphics type does have a relation to point sizes. I made charts comparing type in Geni units to their corresponding point sizes in order to determine what size would be readable, yet small enough to allow room for sufficient flexibility (see fig. 6). I chose 2.0 Geni units or 18 point type. This information then helped me in the construction of a grid to use as an aid in producing layouts (see fig. 7.) I superimposed my grid on the Geni unit grid to determine the coordinates of the lines. The coordinates were then input on the Genigraphics through the keyboard to ensure accuracy. The grid was saved as one file and each of the other elements

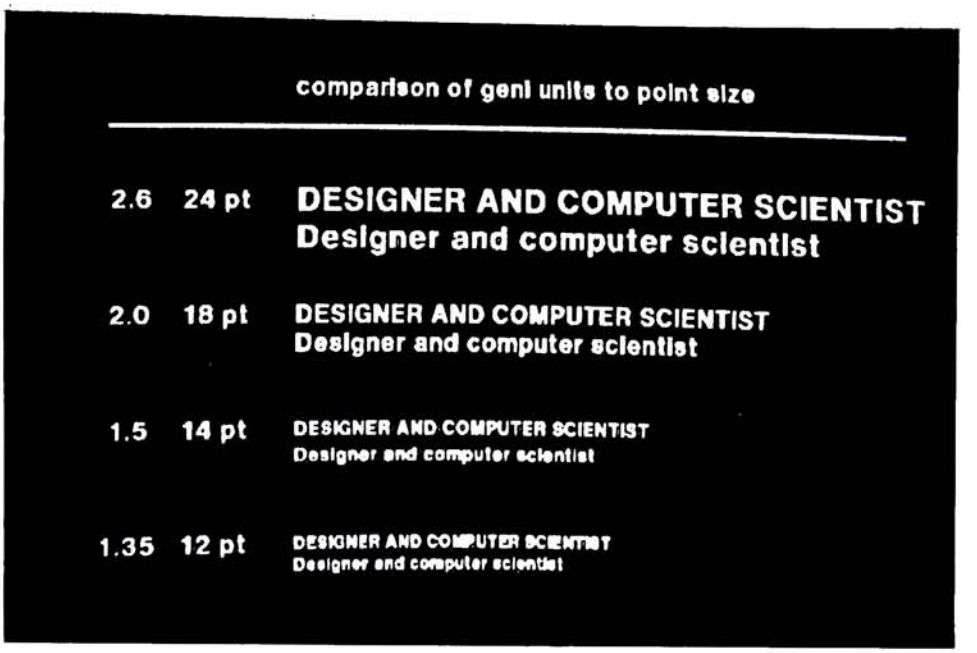


Fig. 6: Type Chart

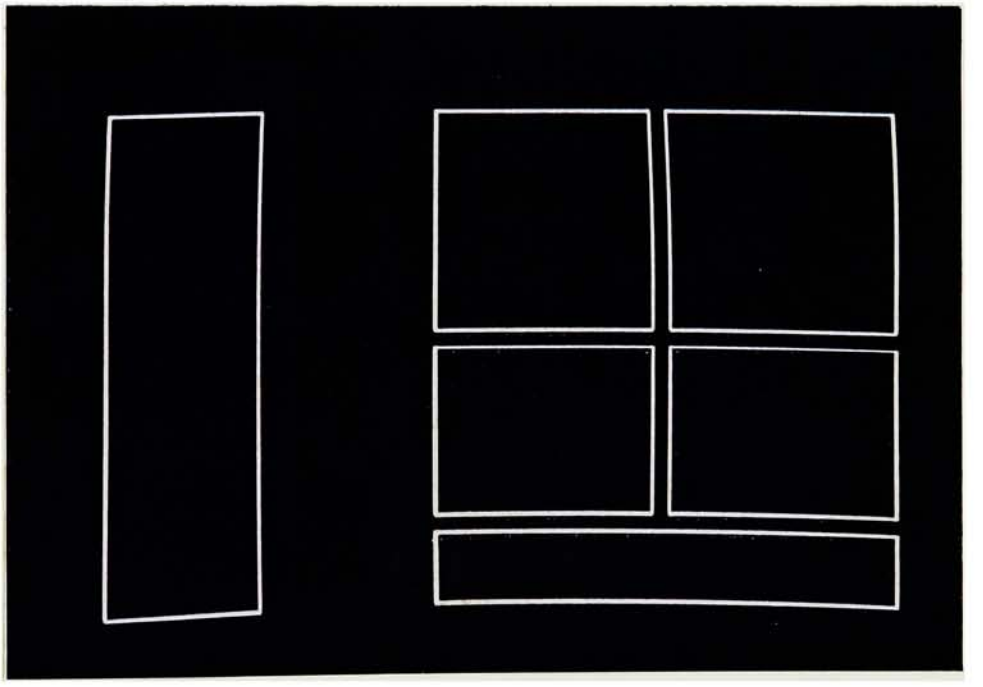


Fig. 7: Grid

was also saved on a disk as separate files. To experiment with layouts, I would simply call up the grid and the other files that held the symbols or images that I wished to use. I entered text through the keyboard using the TXT function which permitted me to specify the size, leading, color, and position of the text. The Genigraphics allowed me to easily and quickly change the position, size, color, and proportions of all the elements within the grid.

The grid then served as my connection between the monitor and the paper sketches. Unfortunately, if I did not stick rigidly to the grid, changes in the layout became difficult to translate. I also could not precisely translate changes I made on the monitor in the grid, symbols, and other objects.

Layout

The layout of the program evolved slowly from its functional needs. I began to experiment with layouts by trying to determine the information that was essential for each frame. This was very dependent on the structure of the program because the

way in which the modules of information were connected changed the information that was necessary. For example, if it was possible to go to any frame in the program from any other frame, then it would be necessary to have a display that would allow you to choose from hundreds of frames. That would have required devising a way of representing each frame that existed in the program. Thus the layout and the structure were directly related and influenced each other. Both changed as I explored different ways of structuring and laying out the program.

There were two general things that needed to be displayed on the monitor: the information that I wished to present and the information about that information or meta-data. I divided the monitor into an image area for text and images and a menu area for symbols and type which represented the information contained in the program. The image area was to be used for viewing information and the menu for selecting options.

Initially, the menu held all options and the image area simply displayed the information which the user had chosen. The menu held the symbols for each face and the four options under that face. It also

displayed arrows that allowed the user to advance or backup frame by frame through an option. The image area was divided into two segments. One was for the information selected and the other to display information to remind the user of where he was in the program.

I explored different methods of dividing the menu and the image area. I decided that a vertical break-up of the screen would provide the most economy of space. The menu used approximately 1/4 of the screen and the image area 3/4. Whether to place the menu on the right or the left side of the screen was the next problem. Arguments could be made for either side and it even raised the question of designing right and lefthand systems. I ultimately chose to put the menu on the left hand side because of the fact that our eye usually goes to the upper left hand corner of a page. We are accustomed to reading from left to right. My rationale was that the menu was the most important thing to see first because it displayed the options that you used to run the program. I further separated the menu and the image area by using different colors for each space.

I tried many different ways of displaying

information in the image area. At first, I used two columns to display type but I found that the type was too small and too hard to read (see fig. 8). I went to one column of type that used both columns of a narrower grid. The first information frames were either all text or image. I later decided to combine images with text to create more visual interest and as an aid in communicating the information.

The background color of the image area changed with the selection of the major options. It became the color of the symbol that was selected as a device to reinforce the structure of the program and to let the user know where he was (see fig. 9). The text was a very light shade of the background color that appeared slightly off-white. White is a very harsh color to look at on monitors and the off-white text was easier to read.

The upper segment of the image area was separated with a ruled line because it served a different function. A symbol of the face that was selected appeared in this segment. I also added a frame counter which told the number of the present frame out of the total number of frames under the option. The menu first displayed all of the symbols.

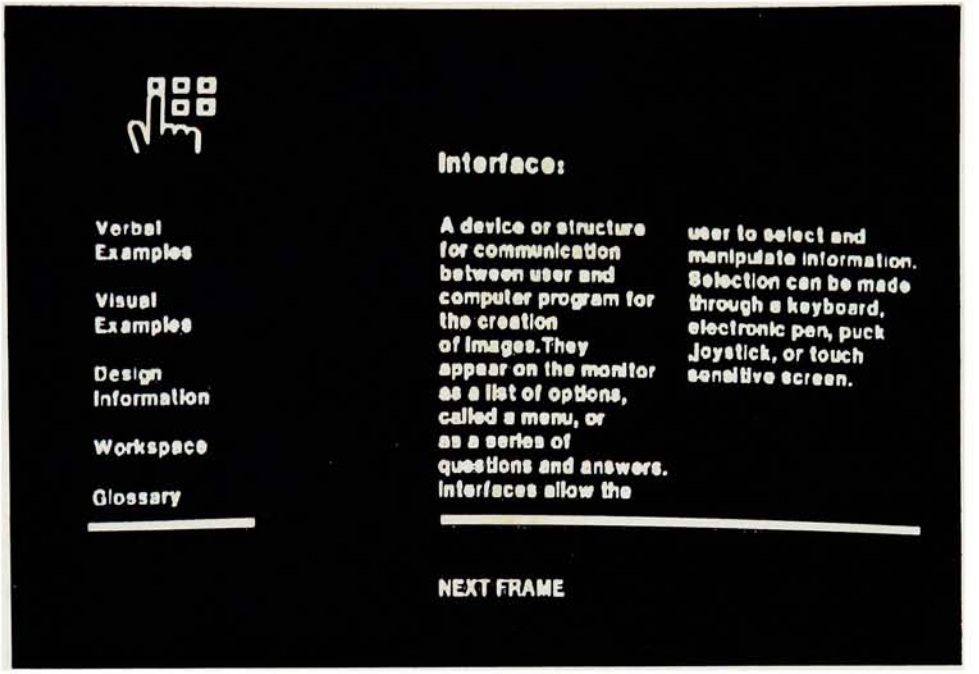


Fig. 8: Two Column Layout

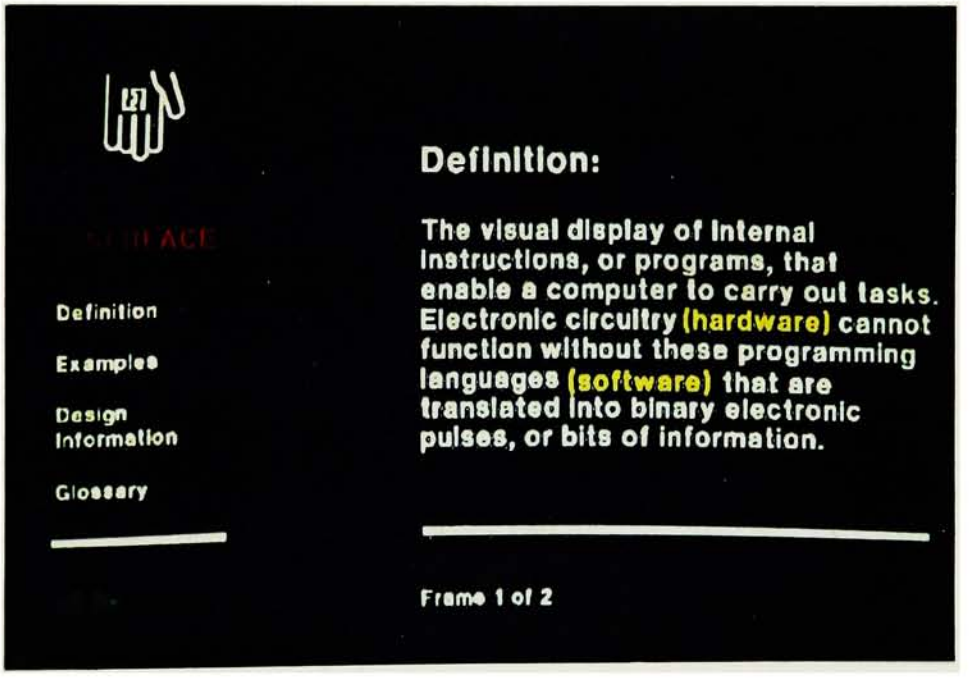


Fig.:9: Color Layout

Next it would display only the symbol chosen and the option under that symbol. Then the menu would change back to the original menu and repeat the process.

I thought that the structure had to be simplified. I also needed to develop a symbol for the information option. This would provide four major options with an additional four sub-options.

A hand combined with a question mark was used for the information symbol. The hand established continuity between all of the symbols. The color of each symbol was based on its conceptual definition. For example, Innerface was blue because it was the furthest removed from human interaction.

Once all four symbols were developed I changed the structure of the program. I was then able to have the menu remain more constant. The symbols always appeared in the upper segment of the menu. The four options of definition, examples, design information, and glossary, were the same for each of the three faces. The options under information were the only exception and consisted of introduction, background, purpose, and glossary. A ruled line separated the sub-options from the main options. Arrows for single frame movement appeared beneath the

sub-options at the bottom of the frame (see fig. 10).

I tested several different colors for the menu before I selected a neutral gray background. The gray background permitted the symbols to be in color, which helped to differentiate them from one another. In addition to this it also helped to lessen the visual competion between the menu and the image area (see fig. 11).

Symbols were incorporated into the menu for several reasons. They are easier to recognize and function better than word descriptions in a menu environment. Good symbols can convey the meaning of a complex concept quickly. "A picture is worth a thousand words." A user can quickly look at a symbol and select it instead of reading a description from the monitor. It is better to keep the amount of reading to a minimum because the flickering nature of monitors can bring fatigue to the eyes. Symbol menus with stylus selection are the way of the future.

An example of the way that the program was to work began with turning the computer on or booting it. A frame would come up with the menu and image area (see fig. 12). Instructions in the image area explained to the user how to proceed. If you were

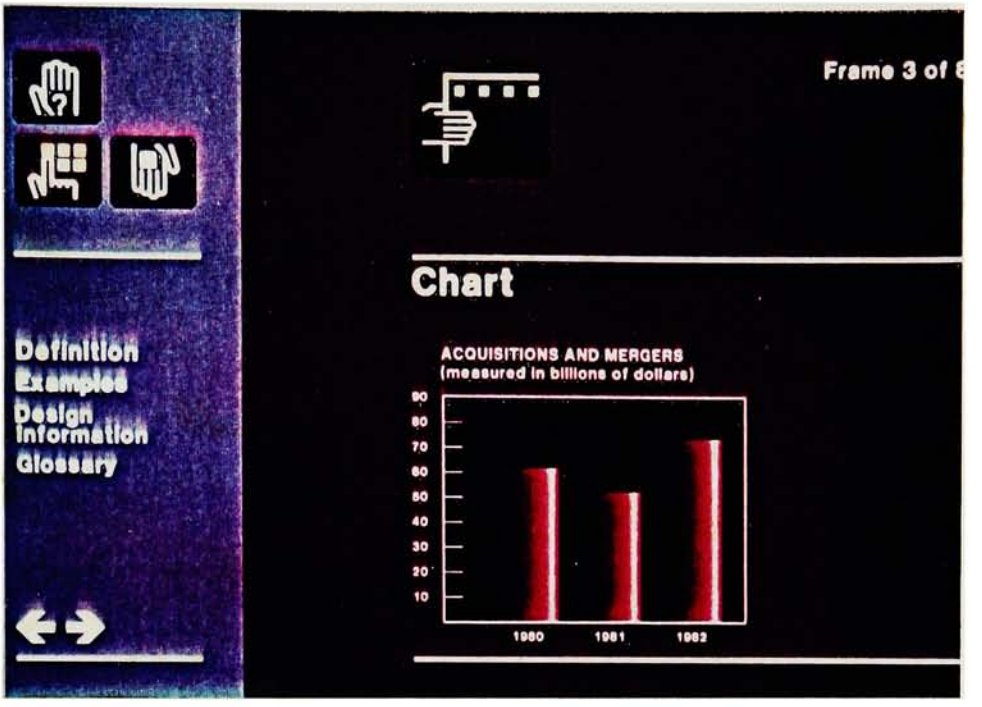


Fig. 10: Prototype Layout

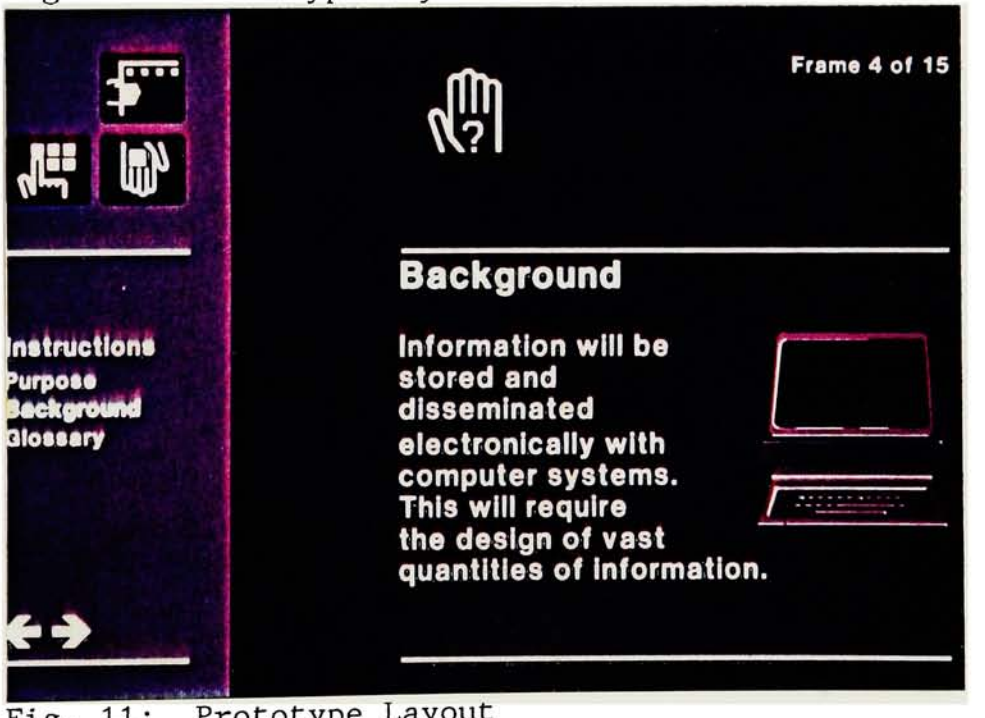


Fig. 11: Prototype Layout

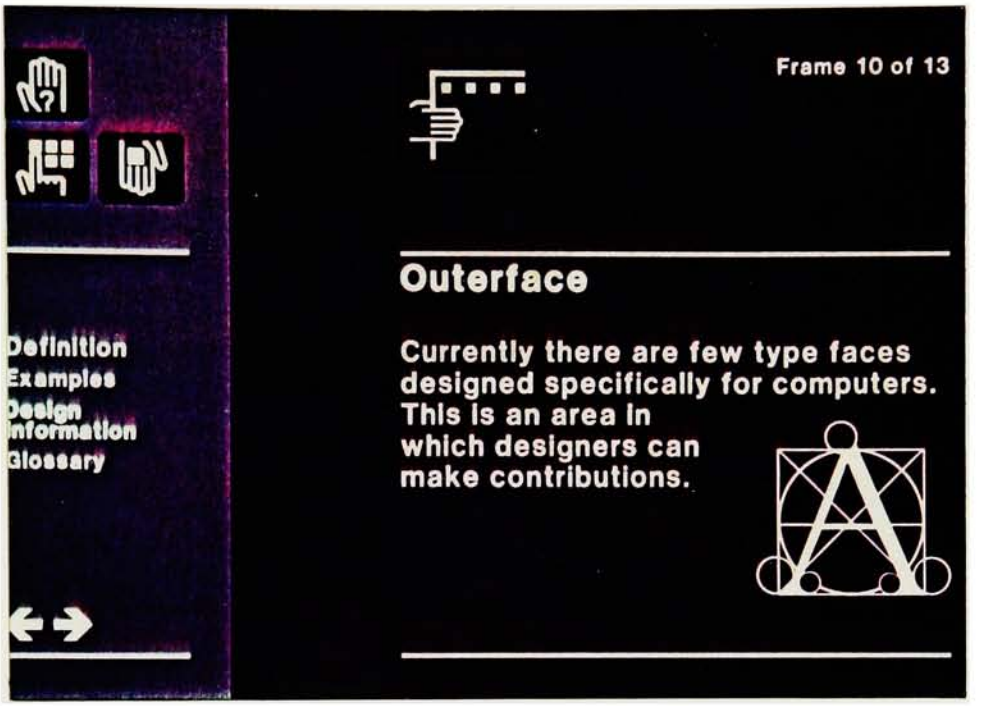


Fig. 12: Prototype Layout

already familiar with the program you would simply select one of the four major options in the upper lefthand corner of the menu. This would be accomplished by moving the cursor to one of the symbols and selecting it. Once the option had been selected that symbol would appear in the upper segment of the image area with the frame counter to remind you of what had been selected and what frame you were in. You would then be free to choose one of the four options under that symbol or choose one of the other three symbols. After selecting one of the four options, the word in the menu which had been selected would be highlighted. The first frame of information on that option would appear in the image area with a descriptive title. You could move through all frames of that option by selecting the advance arrow at the bottom of the menu. Of course, it would always be possible to select one of the other options under that symbol or to select one of the other three symbols.

Adapting the Original Vision

While I worked out the visual details of the program, Gary was busy writing the general code and taking care of hardware problems. We encountered several problems with the DEC equipment that slowed our progress. The DEC 350's were not installed and running until the middle of spring quarter. Also, They did not have as much software and accessories as we first believed. DEC had not yet designed a color monitor for the system and it did not have a graphics tablet. There were also no alternate type fonts although it was possible to change the size of type in the graphics mode.

Gary began programming on a Gigi terminal that was running on the VAX. A core graphics package was being developed for the VAX at the same time which meant that our graphics capabilities were dependent on the progress of the core package. The core package is based on the proposed graphics standards software for standard device independent software, which is supported by ACM SIGGRAPH.

One of the first steps in getting the program running was to input the symbols and store them as

data files. It was necessary to write some code in order to input the symbols and have them displayed on the screen, as well as to manipulate them. Several different ways of inputting the symbols were discussed. Initially, I was going to construct a grid for the symbols that related to the resolution of the monitor. I would then be able to input the coordinates of the points that made up the symbols. However, this approach caused some problems because the symbols contained curvilinear shapes. It would have been difficult to define the coordinates of the curves and it also caused a problem of having too many points. We then decided to digitize the symbols on a Ramtek using a graphics tablet, but it still did not solve the problem with curvilinear shapes. At this time I was also having problems trying to reduce the symbols to a small size without their breaking down.

These problems prompted me to simplify the symbols by using straight line segments and defining the shapes with as few points as possible. The line width of the symbols was increased to make them bolder so that they would hold up at smaller sizes. Although these changes were forced on me, the symbols

were actually much stronger visually afterward. The functional requirements of the technology had enabled me to push the symbols further and make them better (see fig. 13).

I later had to develop a second set of modified symbols to be used in the menu. We found that it would take a long time to regenerate each frame of information because of the number of complex elements. In an attempt to speed up the regeneration of the monitor I further simplified the symbols to line versions. The line versions of the symbols were to be used in the menu where they were small. At that size they were actually better defined with a single line because of the resolution of the monitor.

A problem with switching between computers became obvious after inputting the symbols. When the symbols were called up on the Gigi terminal, they were distorted because of the different aspect ratios of the systems. The aspect ratio is the proportion of the horizontal and vertical dimensions of the picture elements or pixels. For some reason pixels are not square or standardized which causes images to be distorted when they are displayed on different computer systems. This was a major problem in

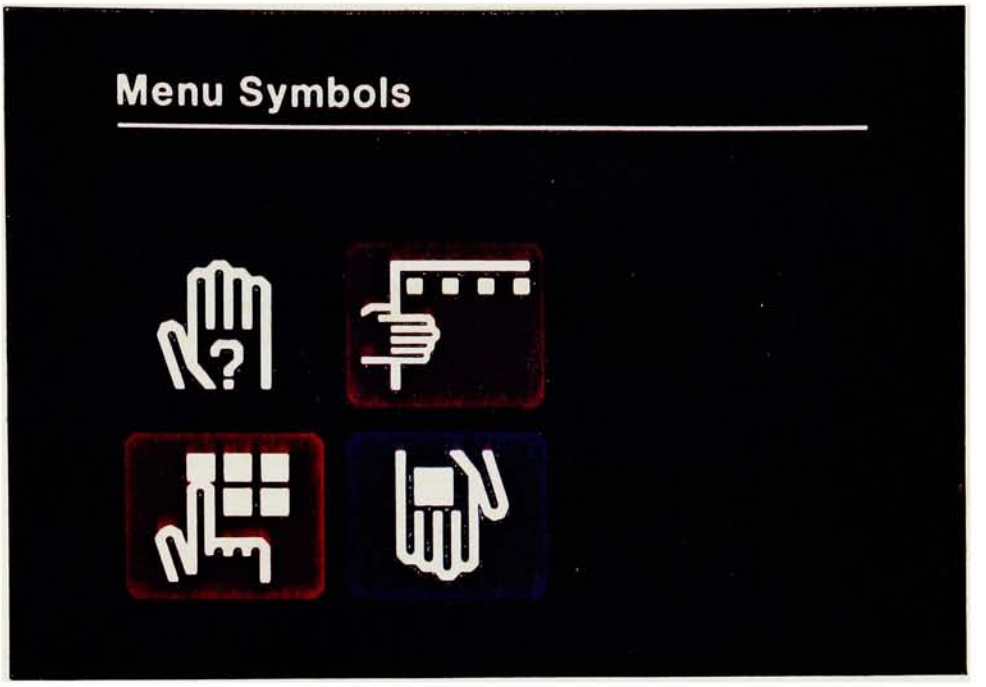


Fig. 13: Simplified Symbols

implementing my layouts for the program because Gary was writing the program to be device independent. Device independence was the major flaw in our approach to the program but we had little choice because the programming had to be started before the PRO 350 was available.

Images were input on a Ramtek and transfered to the Gigi which was connected to the VAX. The PRO 350 was interfaced to the VAX and treated like a VT125. All the different computers involved made it difficult to predict what would happen to images displayed in the final program.

I attempted to compensate for the aspect ratio by devising a grid for the layouts based on the aspect ratio of the Ramtek which was to be used for digitizing the layouts. The initial image would then not be distorted and the additional changes from other systems could be controlled by adjusting the scale for the particular system.

The grid was based on a 649x433 resolution monitor with the correct aspect ratio. It was difficult to make the grid exactly fit the layout of the frames. I had to make adjustments in the layout to fit the units of the grid. Eventually I used a

grid of ten millimeters to the centimeter and enlarged the layout two hundred per cent to position in the 649x433 unit grid. The finer grid allowed me to follow the layout more closely and determine the coordinates of all the elements.

In the end, it turned out to be difficult to use the grid to position objects on the screen. The monitor was divided into four windows or viewports. Each viewport was turned on and off separately to display images in the menu and image area. This breakup of the monitor made it necessary to position the elements for each viewport separately because they each had their own relative set of coordinates and proportions. We had to visually position everything by experimenting with the coordinates of objects in each viewport. This was not the optimum way to produce the layouts for the frames because it defeated the purpose of carefully developing layouts.

The division of the monitor into viewports hindered positioning objects but it helped to speed up the regeneration of the monitor. Each time an option was chosen only the information that changed had to be redrawn instead of the entire screen.

The menu area was divided into two viewports.

The four symbols were in the top segment and the four options in the bottom. The top segment remained constant and the bottom only changed for the Information options which were different than the options under the three Faces. In the original "ideal" program, the symbol that was chosen disappeared from the menu and moved to the upper segment of the image area. The word option that was chosen was highlighted. Having the menu remain constant removed the need for constant regeneration.

The image area was also divided into two segments. One segment contained the symbol of the option chosen with a frame counter and the other held the text or image being displayed. The image segment would have to be regenerated each time but the symbol segment only needed to regenerate when another symbol was chosen.

In order to lay out the individual frames of information I needed to have a method of copyfitting the text. It was important to know how much room the type would take up so that I could plan each frame. I constructed a grid based on the resolution of the PRO 350, which is 960x240. Next I determined the number of pixels that made up the text area. I

determined the number of characters and lines that would fit in the text area based on the 24 lines by 80 columns of type available in the text mode.

Once again these computations turned out to be useless because type is altered in the graphics mode. The size of the type varies with the size of the viewport. The only way to determine the number of characters and lines was to input a sample line of type in each viewport and count the characters.

Another problem that reduced the effectiveness of the layout was that everything had to be reduced to line drawings. A fill routine had not been written for the core graphics package, which limited us to a skeleton of the original layout. I had to use lines to define the menu and image areas instead of solid colors. The symbols and ruled lines were also reduced to outlines.

Color also turned out to be a big problem. The PRO 350 did not come with a color monitor but did have a monochrome monitor capable of displaying eight shades of grey. An attempt was made to interface the 350 to a Barco color monitor, which worked with limited success. It only displayed red, green, blue and black. This was not enough because I needed at

least four colors other than black, one for each symbol. The grey scale for the PRO 350 was also bad because several shades were missing.

The end product that we produced turned out to be a crude skeleton of my original concept. Selections from the menu had to be made through the keyboard because we did not have a graphics tablet. The structure of the program did retain most of the original concept although the visual layout was severely hampered (see appendix 1 for instructions). We were only able to input a limited number of text frames to demonstrate the structure of the program. Color and outline limitations also made it difficult to do anything with the layout beyond indicating the position of objects.

CONCLUSION

In its present form, the program is well below the objectives which I set out to achieve. For many reasons the visual aspects of the program are very crude and the number of frames is limited. It is not an interactive program in the true sense of the word because of the limited actions and responses available to the user. My concept of a flexible, well-designed, highly interactive system was slowly reduced as we encountered hardware and software problems.

Perhaps the basic reason for many of the difficulties I had was that the project is beyond the resources available at RIT. There is not any computer system at RIT which meets the requirements of high resolution, programmability and graphics

tablet input. The Genigraphics has high resolution graphics and a graphics tablet with an electronic stylus but it is not programmable. It also lacks some very important features for graphic design, including: hardcopy output, copyfitting, reliable color standards, video input and updated coordinate information of images on the screen. Genigraphics does have a hardcopy output device but RIT does not possess one. Other systems available are programmable but lack the necessary graphics capabilities.

The DEC PRO 350 that was used caused many problems because it was recently acquired and the bugs were not yet out of the system. Most of the problems arose from a communication problem between the VAX and the PRO 350 computer (see appendix 1). DEC did not have a color monitor for the 350, which necessitated constructing a kluge with a Barco monitor. The color monitor did not work properly and restricted the available colors. A graphics tablet is also not available with the 350, so all interaction was restricted to the keyboard.

Software was also a problem with the DEC equipment because it was just put on the market and

there was not very much software available. Part of the reason why the 350 had to run on the VAX was that C, the language Gary was using to write the program, was not available on the 350. We also did not have access to the 350 until well into the thesis project, so the programming had to be started on the VAX. The core graphics program was not fully developed which restricted the graphics capabilities of the program.

Ultimately the device independent approach caused many of the problems that we encountered. The number of computer systems involved in the programming, graphics and running of the program made it difficult to control the visual display of images. One of the biggest problems with computer systems today is the variation in aspect ratios. Although it is possible to compensate for a known aspect ratio, it becomes difficult to adjust an image when several computers are involved. Non-square pixels effectively make it extremely difficult to design graphics for device independent systems. Square pixels would easily solve this problem. There is no relationship between pixels and standard units of measure, which further compounds problems in transferring images between paper and monitor. I

also found that while designing on the computer it was very difficult to make decisions about the layout without being able to compare different solutions. A hardcopy output device is a must for any computer graphics system to be used as a graphic design tool.

I also discovered that color standards are another major problem with computer graphics systems. There is no way to accurately reproduce color from a monitor on film or paper. Color also varies from monitor to monitor on different and identical systems. Furthermore, the amount of light in the room where the computer is located affects the color on the monitor. I found this to be a problem while working on the Genigraphics at different times of day. Color layouts produced at night with little light in the room appeared totally different during the day when there was much more ambient light. It appears that a controlled environment is necessary to ensure proper viewing of a computer monitor.

Type was another big disappointment in the program. I set out to improve the readability of monitors by using a well-designed typeface. However, the PRO 350 does not have alternate type font capabilities so we were forced to use a standard 7X9

dot matrix font that is native to the 350. Besides the sheer ugliness and illegibility of the font, I found that there was no relationship between the dot matrix characters and standard type sizes or measurements. This is a problem with most computer systems that needs to be solved before graphic designers can accurately design layouts for monitors.

Clearly, many of the problems I encountered are problems inherent in most computer systems. They must be addressed before an effective tool for graphic designers can be created and displays on monitors can be improved. These problems suggest a need for an RIT computer system with high resolution graphics that is programmable. Such a system would have tremendous potential as an educational tool as well as a medium for graphic designers. It could be programmed to run interactive programs and be used for research by students. A programmable system with high resolution graphics would allow the designer to program the system to fit the specific needs of a project rather than adapting a project to a system. There will obviously be an increasing number of students interested in working on projects similar to interactive programs and it would be a great

advantage for RIT to posess the appropriate equipment to carry them out. Perhaps the development of such a system should be one of the first priorities of the new computer graphics major or of individuals interested in furthering computer graphics at RIT.

One possible solution to this problem would be to acquire a hard disk for the Genigraphics system, which operates on a PDP-11/34. A core graphics package could then be programmed on the disk which could be interchanged with the Genigraphics disk. The regular Genigraphics class would not be disturbed and research would be able to take place. A second Genigraphics system may be necessary for this solution depending on the amount of computer time available. This has the potential to be an economical action that would place RIT at the forefront of graphic design/computer graphics research. Undoubtedly the existence of this equipment would recruit students of higher quality to the graphic design and computer graphics graduate programs. By higher quality, I mean students who already posess the graphic design background and skills that are necessary to undertake true thesis research.

Aside from the technical difficulties, the element of time was a major obstacle in completing the thesis project. I began my thesis work two quarters prior to the actual thesis quarter as well as tailoring my other classwork to the thesis project. There simply is not sufficient time to complete a complex technical design project within the limitations of the current credit hour structure. Computer-oriented design projects require research of highly technical data, acquisition or construction of the proper equipment, writing and debugging of complex programs, and the design of layouts and structures of information. In the past the design of layouts would have been a thesis project in itself. New technologies are placing greater demands on the graphic designer because it is necessary to understand and solve problems of a totally different nature from printed material. This is in addition to the aspects of a problem which are the traditional domain of the designer. The number and complexity of technologically oriented design projects will be increasing out of necessity from the needs of society, which will magnify this problem in the coming years.

Although the final product of my thesis project does not meet the objectives that I set out to achieve, I do not consider it a failure. I feel the program was successful in many ways considering that this was the first project of its kind at RIT and that resources were very limited. It is as good as it possibly could be within the limitations of time, hardware and software. The prototype program retains the original concept of a kinetic program structure and serves to demonstrate the fundamental elements of an interactive program. Gary and I have laid the foundation of visual, hardware and software structures for future work in this area. The program was written to be expandable so that when the core graphics program is completed the graphics of the program can be improved and more frames of information added. It is my desire that eventually the program will be able to be used as a tutorial for students in the graphic design department. There is also the possibility of adapting the structure of the program to other informational needs in a learning environment on campus.

Perhaps the most important achievement of this project has been to create an awareness of the

potential and needs of graphic design in computer graphics. This will serve as a catalyst for future joint research by graphic design and computer science students.

SELECTED BIBLIOGRAPHY

De Neve, Rose. "Aaron Marcus: Communicating with Computers." Print, vol. 26, no. 3, June/July 1972, pp. 62-67.

Gerstner, karl. Designing Programmes. New York: Hastings House, 1964.

Kepes, Georgy, ed. Structure in Art and Science. New York: George Braziller, 1965.

————, ed. The New Landscape in Art and Science. Chicago: Paul Theobald and Co., 1967.

Marcus, Aaron. "A Prototypical Computerized Page Design System." Visible Language, vol. 5, no. 3, Summer 1971, pp. 196-220.

————. " At The Edge of Meaning." Visible Language, vol. 11, no. 2, Spring 1977, pp. 9-10.

————. "Graphic Design and Computer Design: Know Buisness is Show Buisness." Centerline, July 1981, pp. 6-9.

————. "When Designing Computer Graphics, The Know Buisness is Show Buisness." Industrial Arts Magazine, March/April 1982, pp. 24-27.

Stankowski, Anton. Visible Presentation of Invisible Processes. Teufen: Arthur Niggli ltd., Sa.

APPENDIX 1

Summary Report

Exploring the need for graphic design in information graphics:

A Summary

Gary A. Roshak
May 20, 1983

As originally envisioned, this project is significantly ahead of its time; this serves to emphasize the lack of understanding in this area and the need for tools of this kind. The current implementation, although incomplete with respect to the vision, serves well to demonstrate the fundamental concepts and has provided an opportunity to explore the current edge of technology in this area. Following is a brief summary of what currently exists, some of the problems encountered, and possible directions for the future.

The current release of the program is a working prototype of the proposed system. As such, it represents the dynamic, rather fluid nature of the original concept: constantly changing and evolving over time. It shows the functional characteristics of the complete system, while at the same time providing a brief glimpse of the underlying concepts embodied in the content. A consistent layout has been used throughout, with attention shown (to the extent possible) to the various visual aspects: color, typography, spacing, layout and symbols. A user can now sit down with a terminal and casually view frames of information in the various areas, moving at one's leisure from one area to another. The number of frames of information present is currently limited, but these can be very easily added at another time.

A major source of problems throughout was hardware. Much time was taken up exploring the possibility of using one of the personal computers as a vehicle for the system. Once this decision was reached, actually trying to get hold of the equipment and getting it operating was yet another problem. There are still problems with using the machine, and among them are:

1. Color selection. The current choice of colors is bad, being limited to the four primary colors (red, green, blue and black) and not very good shades of those. Gray scale is also bad with several colors missing and the distinction between the visible ones very slight.

2. Communications. The Pro/350 does not interface well with the VAX system running Unix(tm). Running at low baud rates and using the Barco monitor the system will work, albeit grudgingly. At times the machine seems to drift off somewhere, not finishing the picture it is drawing nor responding to commands. It would appear that the Pro/350 is sending handshake characters at high baud rates, which Unix is either not recognizing or not responding to in the expected manner.

Essentially, the problems boil down to the fact that the Pro/350 is not quite a VT125, though we are treating it as such. Two possible approaches for resolving these problems are: First, getting a C compiler running on the VMS systems and then moving our system and the CORE package over to VMS. Hopefully the Pro/350 could then communicate more readily with the VMS VAX. Alternatively, and perhaps more appropriate, would be to get the application running in native mode on the Pro/350. This would entail getting the Whitesmiths C compiler for the Pro/350, but then moving the application and the CORE package would be very easy.

In terms of the future, much can be done with this application. We have simply scratched the surface of our original vision, in the process proving its validity. At the top of the wish list should be implementing this system on the Genigraphics machine. If at some future point these people become more cooperative regarding access to the images generated on that machine, this would be the ideal direction to pursue. In the meantime, we must make do with what is at hand.

The current system only allows for the display of various text frames, and thus does not allow for any interactive graphics. A primary goal for any additional release of the system should implement interactive visual images (both generation and display) to exemplify the graphic design concepts involved here (i.e. option 2 on the menu.)

Another area to be looked at is the internal trapping and handling of error messages. The current implementation has limited error handling capabilities built in, but does nothing with errors reported from the CORE package.

The number of frames currently existing is also very limited. These frames are easily added, existing as simple editor files, and thus could be input by most anyone. However, in the next release of the system the underlying mechanism for these frames should be changed. Currently all frames are created during initialization processing. Moving from one frame to another thus simply involves turning on or off the appropriate segment. This approach works fine for the small number of segments that currently exists. As this number grows, however, the feasibility of this approach deteriorates. An alternative method, roughly speaking, would be as such: During initialization processing, approximately 30 frames would be created and retained permanently. These frames would consist of the border, various menu pieces, large symbols and banners for each face and their accompanying page numbers, and the first frame for each option under each face. This list represents those frames that are involved in moving about through the system, and thus that must always exist since you can move to any of these places at any time. During the actual viewing of frames of information, the system can be dynamically creating and deleting segments while the user is reading a frame. In this manner the system would always stay ahead of the viewer by one or two frames in either direction, and the total number of segments existing or needing to be created at any one time would be kept to a minimum.

These are but a few of the possible improvements to the current prototype system. The possibilities are endless... (using a tablet and mouse for instance, ala the Apple Lisa or Genigraphics.) It is my sincere hope that any successors on this project find it as interesting and rewarding to work on as I have.

INSTRUCTIONS

To initiate the application, enter the command "demo" from the shell (within the proper directory). The message "Please wait" will appear on the screen while the various segments are being built. After a short delay, the Instructions menu will appear on the screen along with the Instructions symbol.

The four areas that can be explored are:

A - Instructions
B - Outerface
C - Interface
D - Innerface

At this point the user should choose one of the options under Instructions, or choose another area to view by typing the corresponding letter (all commands must be followed by a return.) If another area is chosen, the Instructions menu will be replaced by the standard menu.

From any point in the tutorial the user may:

- choose another option under the current face, by entering the number of that option from the menu.
- choose another face to explore, by entering the letter of that face from the menu.
- advance to the next frame, by entering a ">" or just a return.
- go back to the previous frame, by entering a "<".
- terminate the application, by entering a "Q" or "q"

Advancing beyond the first or last frame within an option simply returns you to the symbol for that face. These symbols serve as the entry point for each face.

If an error should occur while trying to open a file, a message will appear on the screen and the program will pause. Simply hit return and control will be returned to the shell.

File conventions:

1. Files with a ".dat" suffix contain data points for the large (hollow) symbols.
   - inner.dat
   - inter.dat
   - outer.dat
   - instr.dat

2. Files with a ".lin" suffix contain data points for the small line drawing symbols.
   - inner.lin
   - inter.lin
   - outer.lin
   - instr.lin

3. Files with a ".txt" suffix contain information for text frames. Each frame is contained in a separate file, with names taking the form "FACEn.txt" - i.e. inner4.txt

Data Structures:

Data points are stored as a structure in binary form, where the structure contains:

1. a code indicating moveto (m) or lineto (l)
2. the x & y coordinate of a point as a real number

Text frames are stored in character format, each frame in a separate file. The structure of the file is:

1. the first line contains the frame offset, used to create the segment number, followed by a carriage return
2. the second line contains the page number message, i.e. "Frame 1 of 5"
3. the remainder of the file contains lines of text exactly as they will appear in the frame

Segments used:

1 - border
2 - menu symbols
3 - menu options
4 - instruction options
9 - "Please Wait"
1000 - large Innerface symbol
1500 - page number for symbol
1999 - banner for symbol
2000 - large Interface symbol
2500 - page number for symbol
2999 - banner for symbol
3000 - large Outerface symbol
3500 - page number for symbol
3999 - banner for symbol
4000 - large Instructions symbol
4500 - page number for symbol
4999 - banner for symbol
32767 - error message

1001 - 1006 - text frames
2001 - 2006 - " "
3001 - 3006 - " "
4001 - 4006 - " "

1501 - 1506 - page numbers for text frames
2501 - 2506 - " " " " "
3501 - 3506 - " " " " "
4501 - 4506 - " " " " "

PURPOSE

Designer and computer graphics consultant Aaron Marcus has developed a concept that computer systems can be divided into three components or faces. The three faces are the Innerface, Interface, and Outerface.

Each face is an area in which designers can become involved to improve the electronic communication of information.

The purpose of this program is to explain the three faces of computers to provide designers with a conceptual framework to approach these new tools.

It is also intended to provide an example of graphic design principles applied to the design of each face and to promote computer literacy in the design field.

The program is not intended to be an exhaustive source of information on computers and contains very little technical information. It is an introduction to computers and their relationship with the field of graphic design.

Information is presented to point out areas where design skills are needed and to encourage creative people to become involved in the design and use of the new technology.

BACKGROUND

The graphic design field is faced with revolutionary changes in both the tools it uses and the products it designs.

Computer technology is changing our society from an industrial one to an information-based one. This shift in technology has many implications for the design field.

The quantity and complexity of information is growing at a tremendous rate which magnifies the need for effective communication.

Information will be stored and disseminated electronically through computer systems. This will require the design of vast quantities of information.

Designers will be designing information that may never appear in printed form but exist solely as electronic information.

They will also have to apply their expertise to the design of computer systems and the way in which people communicate with them. It is not enough just to have information. There must be an easy way to find and use it.

Information graphics are the greatest responsibility and challenge for designers in the computer age.

As the complexity of information that the designer must communicate increases, the need for faster, better tools will become a necessity.

Computer graphics systems can be powerful design because of their speed and flexibility. It is possible to manipulate images and generate alternative designs very fast.

Computers can free the creative person from the drudgery of manually drawing out ideas. It is possible to create more sketches in a shorter amount of time which will undoubtedly lead to better solutions and increased productivity.

It is important to understand that computers only carry out instructions from users. Computers cannot differentiate between good and bad design, yet.

Computers are simply tools that designers can use like a t-square or triangle only much more powerful. Design is design whether you use a pencil or an electronic stylus.

Computers may require that the designer have a better understanding of the systematic approach to creative problem-solving which combines logic and intuition.

This will be necessary because the designer will be faced with many decisions more frequently and it will be difficult to make these decisions without an orderly thought process.

Although computer graphics systems have great potential, most systems currently lack essential features for layout work, and communication is often awkward. This situation is changing and will change even more as designers become involved in the new technology.

An outerface is the output or end product of communication with the computer. In computer graphics systems the outerface is an image created through interfaces. All computers are in a sense computer graphics systems because they display information in some form.

Computers can generate many types of images including charts, graphs, text, maps, photos, symbols, animation, and illustration. Images can be displayed on a screen, either the computer's monitor or television screen, on paper from printers and plotters, or on film for slides and animation.

KEY WORDS
Hardcopy
Printout
Printer
Plotter
Slide
Film
Artwork
Output
Post Process

OUTER INFO

Outerfaces are traditional design problems such as page layouts and graphs that can be effectively solved on the computer.
The resolution (fineness or coarseness of the dot pattern) of the computer monitor and of the hardcopy output device are important considerations when designing Outerfaces. They have an effect on the sophistication of images that can be used.

Low resolution monitors require simple shapes and large type because the image quality is poor. Curvilinear forms appear jagged and it is difficult to define small objects.

High resolution monitors can display complex geometric and organic forms with better image quality.

Printers generally have relatively low resolution output. Plotters can render high resolution line drawings. Both have varying color capabilities depending on the type of system.

Capturing the image on film is a common way of obtaining high resolution output. This can be done by shooting directly from the monitor with a camera or through the use of a film recording device.

Images can also be stored on a magnetic disk or tape for later reproduction.

Resolution is also important when choosing type size and style. Type faces often look awkward because the image quality of the monitor is not high enough for small type or finely detailed letterforms.
Currently there are few type faces designed specifically for computers. This is an area in which designers can make contributions.

Different computer systems also have varying color capabilities from one to sixteen million depending on the sophistication of the system.

Designers also have another important task with Outerfaces. They can provide standards and formats that can be built into software which enables computers to design images automatically or semi-automatically.

In the future there will be so much information that it will be impossible for designers to design it all. It will be necessary to transfer some of the designer's expertise to the computer.

An Interface is a device or structure for communication between a user and computer program for the creation of images. It appears on the monitor as a list of options, called a menu, or as a series of questions and answers.

Interfaces allow the user to select and manipulate information. Depending on the type of computer system, selection can be made through a keyboard, electronic pen, puck, joystick, touch sensitive screen, or voice input.

KEY WORDS

Menu
Joystick
Monitor
Electronic Pen
Stylus
Puck
Mouse
Keyboard
Voice Input
Cursor

INTER INFO

Interfaces are perhaps the most important aspect of a computer because they stand between the user and the power of a computer. The speed and versatility of computers are useless if communication with them is difficult.

Designers must become involved with Interfaces to make them more natural and easier to use. They can do this by improving the visual and mechanical communication devices such as menus, tablet overlays, and keyboard overlays.

There has been a lack of design principles applied to information displayed on computer monitors. This is also true of the documentation and instructions that accompany these systems.

The traditional design considerations that are used in composing a printed page should be applied to information on the monitor.

Composition, type size, column width, color, and negative space are some examples of these principles.

Menus, the display of information and instructions, are a very important concern of designers. They permit the user to select and manipulate information.

Menus can be made more effective through visu!l improvements that enhance communication. The use of symbols, text, color, placement, and other design elements can make it faster and easier to select options and manipulate information.

Designers can also create the conceptual structures of information. Computers are essentially non-linear books which let the user jump around in information at different levels.

Computers have tree-like structures that brach out into different modules of information. Graphic design can visually reinforce these structures by making them more apparent so it is easier to move through information.

There are three important pieces of information that should appear in each image on the monitor. As a user, you must be able to discern where you are in the information structure, where you have been, and

what options are available next. These visual clues will give the user a sense of orientation while moving through complex information.

An Innerface is the visual display of internal instructions, or programs, that enables a computer to carry out tasks.

Electronic circuitry (hardware) cannot function without these programming languages (software) that are translated into binary electronic pulses, or bits of information.

There are different levels of programming languages, from machine and assembly languages to high level programming languages. These languages communicate with the computer at different levels of abstraction.

KEY WORDS

Pascal
Fortran
Basic
Cobol
C
Assembly Language
Machine Language
Operating System
System Software
Software

NER INFO

An important potential design area is the visual display of programs or innerfaces. The design of this information that controls the computer has been ignored until recently.

Computers and programs are becoming more complex which will necessitate simplifying instructions and their display if programmers are to work efficiently.

The programs that run the computers of large corporations and government agencies are very complex. When the programmers who write these programs are no longer thereit can become a difficult task to decipher their programs.

Designers can make it easier for computer scientists to read, understand, and debug programs by improving their visual format.

Today it is often very difficult for a programmer to quickly understand the function of a program at a glance. Through the use of symbols and improved layout, it is possible to communicate the meaning of a program more easily.

Color can also be an effective way of seperating blocks or functions of a program. It is also important to make programs self-documenting by choosing words that describe their actions.

APPENDIX 3

Program Code

```
define XMIN     0.0
define XMAX     100.0
define YMIN     0.0
define YMAX     100.0

ypedef struct
        {
        char code;
        FLOAT x, y;
        } POINT;

har surfname[] = "TERM";
loat sx, sy, tx, ty;
xtern int currseg, currface, currbanner, currmenu;

*****************************************************************************
*
*       control program for demo system
*
*****************************************************************************/


CID main(argc, argv)
        int argc;
        char *argv[];
    {
    long leveln;
    char color;
        int segname;

  initializecore("tuffered", "no-input", "2d");
    initializeviewsurface(surfname);
    selectviewsurface(surfname);

        initialize();
        select();

    deselectviewsurface(surfname);
    terminateviewsurface(surfname);
    terminatecore();
        exit(0);

    }

/*****************************************************************************
*
*       initialize:             initialization processing - creates all segments
*                                                   and displays initial segments
*
*****************************************************************************/

ICID initialize()
        {

        setvisibility(OFF);

        border();
        menusymbols();
        instrmenu();
        menu();
        faces();
        banner();
        createinstr();
        createouter();
```

```
    createinter();
    createinner();
    currseg = currface = INSTR;
    currbanner = INSTR + BANNEROFFSET;
    currmenu = INSTRMENU;

    beginbatchofupdates();
            setsegmentvisibility(INSTRMENU,ON);
            setsegmentvisibility(currseg + PAGEOFFSET,ON);
            setsegmentvisibility(currbanner,ON);
            setsegmentvisibility(WAIT,OFF);
            setsegmentvisibility(currseg,ON);
    endbatchofupdates();

    }       /* end initialize() */


*****************************************************************/

    draw:   read data points from file, and then display them
                    data points are store as binary information with a
                    code indicating "move to" or "line to"

*****************************************************************/

D draw(file)
    int file;       /*      file descriptor */
    {
    POINT datum, *pd;
    char s[BUFSIZ];

    pd = &datum;
    while ((read(file, pd, sizeof(POINT))) > 0)
            {
            scale(sx,sy,pd);
            translate(tx,ty,pd);
            if (pd->code == 'm')
                    moveabs2(pd->x, pd->y);
            else if (pd->code == 'l')
                    lineabs2(pd->x, pd->y);
            }

    }

*****************************************************************

 cursorhome:  moves cursor to bottom left corner of screen

*****************************************************************

D cursorhome()
    {
  setviewport2(0.0, 1.0, 0.0, 1.0);
  setwindow(XMIN, XMAX, YMIN, YMAX);
    moveabs2(100.0,50.0);
    }

*****************************************************************

    error:  error handler for files that can't be opened.

            prints a message and exits with an error value.

*****************************************************************/
```

```
error(code)
        char *code;
        {
        char buf[80];

        deleteallretainedsegments();
        cursorhome();
        createretainedsegment(32767);
        moveabs2(XMAX/2,YMAX/2);
        sprintf(buf," can't open file %s ",code);
        text(buf);
        closeretainedsegment(32767);
        setsegmentvisibility(32767,ON);
        getchar();
        terminatecore();
        exit(1);

        }       /* end error() */
/*****************************************************************************
 *
 *      menusymbols:    create permanent segment containing symbols of
 *                                      each face for the menus.
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define MXMIN   0.0
#define MXMAX   650.0
#define MYMIN   0.0
#define MYMAX   980.0

extern char *face[];
float sx, sy, tx, ty;

VOID menusymbols()
    {
    char color;
        int mfd[4], i, segname;


        for (i = 0; i < 4; ++i)
                if ((mfd[i] = open(face[i], 0)) == -1)
                        error(face[i]);

    setviewport2(0.0, 0.3, 0.0, 1.0);
    setwindow(MXMIN, MXMAX, MYMIN, MYMAX);
    setwindowclipping(YES);

        segname = 2;
                createretainedsegment(segname);
                color = 'G';
                setcolor(color);
                setintensity(1.0);

                moveabs2(MXMAX,MYMIN);
                lineabs2(MXMAX,MYMAX);

                sx = 0.3;
                sy = 0.3;

                tx = 125.0;
                ty = 775.0;
                draw(mfd[3]);
                close(mfd[3]);
```

```
                ty = 775.0;
                draw(mfd[2]);
                close(mfd[2]);

                tx = 125.0;
                ty = 625.0;
                draw(mfd[1]);
                close(mfd[1]);

                tx = 350.0;
                ty = 625.0;
                draw(mfd[0]);
                close(mfd[0]);

                moveabs2(75.0,900.0);
                text("A");
                moverel2(0.0,-45.0);
                text("B");

                moveabs2(75.0,750.0);
                text("C");
                moverel2(0.0,-45.0);
                text("D");

                closeretainedsegment();
                setsegmentvisibility(segname,ON);

    }   /* end menusymbols() */
/*****************************************************************************
 *
 *  menu:       create segment containing standard menu options
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define MXMIN   0.0
#define MXMAX   650.0
#define MYMIN   0.0
#define MYMAX   980.0

extern char *face[];
float sx, sy, tx, ty;

VOID menu()
    {
    char color;
        int mfd[4], i, segname;

    setviewport2(0.0, 0.3, 0.0, 1.0);
    setwindow(MXMIN, MXMAX, MYMIN, MYMAX);

        segname = 3;
                createretainedsegment(segname);
                color = 'G';
                setcolor(color);
                setintensity(1.0);

                moveabs2(125.0,575.0);
                lineabs2(550.0,575.0);

                moveabs2(125.0,500.0);
                text("Definitions");
```

```
                text("Design");

                moverel2(0.0,-45.0);
                text("Information");

                moverel2(0.0,-65.0);
                text("Examples");

                moverel2(0.0,-65.0);
                text("Glossary");

                moveabs2(75.0,500.0);
                text("1");
                moverel2(0.0,-65.0);
                text("2");
                moverel2(0.0,-110.0);
                text("3");
                moverel2(0.0,-65.0);
                text("4");

                moveabs2(125.0,130.0);
                text("<=  =>");

                moveabs2(125.0,70.0);
                lineabs2(550.0,70.0);

                closeretainedsegment();
                setsegmentvisibility(segname,OFF);

    }   /* end menu() */
/*****************************************************************************
 *
 *  instrmenu:  create segment containing menu options for instructions
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define MXMIN   0.0
#define MXMAX   650.0
#define MYMIN   0.0
#define MYMAX   980.0

extern char *face[];
float sx, sy, tx, ty;

VOID instrmenu()
    {
    char color;
        int mfd[4], i, segname;


    setviewport2(0.0, 0.3, 0.0, 1.0);
    setwindow(MXMIN, MXMAX, MYMIN, MYMAX);

        segname = 4;
                createretainedsegment(segname);
                color = 'G';
                setcolor(color);
                setintensity(1.0);

                moveabs2(125.0,575.0);
                lineabs2(550.0,575.0);

                moveabs2(125.0,500.0);
```

```
        moverel2(0.0,-65.0);
        text("Instructions");

        moverel2(0.0,-65.0);
        text("Purpose");

        moverel2(0.0,-65.0);
        text("Background");

        moveabs2(75.0,500.0);
        text("1");
        moverel2(0.0,-65.0);
        text("2");
        moverel2(0.0,-65.0);
        text("3");
        moverel2(0.0,-65.0);
        text("4");

        moveabs2(125.0,130.0);
        text("<=  =>");

        moveabs2(125.0,70.0);
        lineabs2(550.0,70.0);

        closeretainedsegment();
        setsegmentvisibility(segname,OFF);

    }   /* end instrmenu() */

/*****************************************************************************
 *
 * border:      draws a border around the entire viewing area, creating
 *              this border as a separate segment (#1).
 *
 *              also creates a segment with wait message for
 *              initialization (#9).
 *
 *****************************************************************************/

#include "defs.h"
#define XMIN    0.0
#define XMAX    100.0
#define YMIN    0.0
#define YMAX    100.0

VOID border()
        {
        int segname;
        char color;

        segname = 1;
        cursorhome();           /* homes cursor; also sets window and viewport *
        createretainedsegment(segname);
        color = 'G';
        setcolor(color);
        setintensity(1.0);
        moveabs2(XMIN,YMIN);
        lineabs2(XMIN,YMAX);
        lineabs2(XMAX,YMAX);
        lineabs2(XMAX,YMIN);
        lineabs2(XMIN,YMIN);
        closeretainedsegment();
        setsegmentvisibility(segname,ON);

        /*      create segment with wait message */
```

```
*
*                       page numbers are segments 1500, 2500, 3500, 4500
**********************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    1100.0
#define YMIN    0.0
#define YMAX    470.0

char *facesym[] = {
        "inner.dat",
        "inter.dat",
        "outer.dat",
        "instr.dat",
                                };
extern float sx, sy, tx, ty;

VOID faces()
    {
    char color;
        int fd[4], i, segname;


        for (i=0; i < 4; ++i)
                if ((fd[i] = open(facesym[i], 0)) == -1)
                        error(facesym[i]);

        sx = sy = 0.2;
        tx = 175.0;
        ty = 70.0;

        for (segname=0; segname < 4; ++segname)
                {

                /* create page number segment */

                setviewport2(0.65, 1.0, 0.60, 1.0);
                setwindow(0.0, 100.0, 0.0, 100.0);

                createretainedsegment((segname + 1) * SEGBASE + PAGEOFFSET);
                        color = 'R';
                        setcolor(color);
                        setintensity(1.0);

                        moveabs2(50.0,80.0);
                        text("Frame 0 of n");

                closeretainedsegment();
                setsegmentvisibility((segname + 1) * SEGBASE + PAGEOFFSET,OFF);

                /* create symbol frame segment */

                setviewport2(0.3, 1.0, 0.0, 0.60);
                setwindow(XMIN, XMAX, YMIN, YMAX);

                createretainedsegment((segname + 1) * SEGBASE);
                        setintensity(1.0);

                        color = 'B';
                        setcolor(color);

                        moveabs2(175.0, 460.0);
                        lineabs2(900.0, 460.0);
```

```
        segname = WAIT;
        createretainedsegment(segname);
        color = 'G';
        setcolor(color);
        setintensity(1.0);
        moveabs2(XMAX/2,YMAX/2);
        text("Please Wait");
        closeretainedsegment();
        setsegmentvisibility(segname,ON);

        }               /* end border() */
/*****************************************************************
 *
 *  banner:     creates frames containing banner symbols for each face
 *              banners are segments 1999, 2999, 3999, 4999
 *
 *****************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    750.0
#define YMIN    0.0
#define YMAX    470.0

extern char *face[];
extern float sx, sy, tx, ty;

VOID banner()
    {
    char color;
        int fd[4], i, segname;


        for (i=0; i < 4; ++i)
                if ((fd[i] = open(face[i], 0)) == -1)
                        error(face[i]);

    setviewport2(0.3, 0.6, 0.6, 1.0);
    setwindow(XMIN, XMAX, YMIN, YMAX);
        sx = sy = 0.45;
        tx = 275.0;
        ty = 180.0;

        for (segname=0; segname < 4; ++segname)
                {
                createretainedsegment((segname + 1) * SEGBASE + BANNEROFFSET);
                setintensity(1.0);

                color = 'R';
                setcolor(color);
                draw(fd[segname]);
                close(fd[segname]);

                closeretainedsegment();
                setsegmentvisibility((segname + 1) * SEGBASE + BANNEROFFSET,OFF);
                }

        }       /* end banner() */

/*****************************************************************
 *
 *  faces:      creates frames containing symbols for each face
 *
```

```
                        {
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = OUTER;
                currbanner = OUTER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case 'C':                       /* interface */
case 'c':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                setsegmentvisibility(currbanner,OFF);
                if (currmenu = INSTRMENU)
                        {
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = INTER;
                currbanner = INTER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case 'D':                       /* innerface */
case 'd':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                setsegmentvisibility(currbanner,OFF);
                if (currmenu = INSTRMENU)
                        {
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = INNER;
                currbanner = INNER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case '1':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                currseg = currface + 1;
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case '>':                       /* next frame */
case '\n':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                ++currseg;
```

```
                moveabs2(175.0, 60.0);
                lineabs2(900.0, 60.0);

                draw(fd[segname]);
                close(fd[segname]);

            closeretainedsegment();
            setsegmentvisibility((segname + 1) * SEGBASE,OFF);
            }

        }       /* end faces() */

/*****************************************************************************
 *
 *      select:         menu driver - reads user's selection from menu and
 *                              takes appropriate action.  Currently it simply t
 *                              the corresponding segments on and off.
 *
 *****************************************************************************/

#include "local.h"
#include <ctype.h>
#include "defs.h"

#define SEGLIMIT        (currface + NUMOFSEGS)
#define NUMOFSEGS       6

extern int currseg, currface, currbanner, currmenu;

VOID select()
        {
        char option[5];

        do
            {
            cursorhome();
            getln(option,5);                    /* read option and carriage retu

            switch (*option)
                {
            case 'A':                           /* instructions */
            case 'a':
                beginbatchofupdates();
                    setsegmentvisibility(currseg,OFF);
                    setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                    setsegmentvisibility(currbanner,OFF);
                    if (currmenu = MENU)
                        {
                        setsegmentvisibility(MENU,OFF);
                        setsegmentvisibility(INSTRMENU,ON);
                        currmenu = INSTRMENU;
                        }
                    currseg = currface = INSTR;
                    currbanner = INSTR + BANNEROFFSET;
                    setsegmentvisibility(currbanner,ON);
                    setsegmentvisibility(currseg,ON);
                    setsegmentvisibility(currseg + PAGEOFFSET,ON);
                endbatchofupdates();
                break;
            case 'B':                           /* outerface */
            case 'b':
                beginbatchofupdates();
                    setsegmentvisibility(currseg,OFF);
                    setsegmentvisibility(currseg + PAGEOFFSET,OFF);
```

```
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = OUTER;
                currbanner = OUTER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case 'C':                               /* interface */
case 'c':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                setsegmentvisibility(currbanner,OFF);
                if (currmenu = INSTRMENU)
                        {
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = INTER;
                currbanner = INTER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case 'D':                               /* innerface */
case 'd':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                setsegmentvisibility(currbanner,OFF);
                if (currmenu = INSTRMENU)
                        {
                        setsegmentvisibility(INSTRMENU,OFF);
                        setsegmentvisibility(MENU,ON);
                        currmenu = MENU;
                        }
                currseg = currface = INNER;
                currbanner = INNER + BANNEROFFSET;
                setsegmentvisibility(currbanner,ON);
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case '1':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                currseg = currface + 1;
                setsegmentvisibility(currseg,ON);
                setsegmentvisibility(currseg + PAGEOFFSET,ON);
        endbatchofupdates();
        break;
case '>':                               /* next frame */
case '\n':
        beginbatchofupdates();
                setsegmentvisibility(currseg,OFF);
                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                ++currseg;
```

```
            if (currseg > SEGLIMIT)
                    {
                    currseg = currface;
                    setsegmentvisibility(currseg,ON);
                    setsegmentvisibility(currseg + PAGEOFFSET,ON);
                    }
            setsegmentvisibility(currseg,ON);
            setsegmentvisibility(currseg + PAGEOFFSET,ON);
    endbatchofupdates();
    break;
  '<':                          /* previous frame */
    beginbatchofupdates();
            setsegmentvisibility(currseg,OFF);
            setsegmentvisibility(currseg + PAGEOFFSET,OFF);
            --currseg;
            if (currseg < currface)
                    {
                    currseg = currface;
                    setsegmentvisibility(currseg,ON);
                    setsegmentvisibility(currseg + PAGEOFFSET,ON);
                    }
            setsegmentvisibility(currseg,ON);
            setsegmentvisibility(currseg + PAGEOFFSET,ON);
    endbatchofupdates();
    break;
  '2':
  '3':
  '4':
ult:
    /*      not currently implemented - therefore NOP
     *
     *      for cases 2,3,4 should turn current seg and page
     *      off, and then turn on seg for first frame of that
     *  area.
     *
     *      default action should remain a NOP, or perhaps
     *  print a warning message.
     */
    break;
    }               /* end switch */

    while ((*option != 'Q') && (*option != 'q'));
            /* Q or q to quit */

    /* end select() */
*****************************************************************

    create text frames and accompanying page number
                    frames for Instrface.

*****************************************************************/
```

```
                                        {
                                        currseg = currface;
                                        setsegmentvisibility(currseg,ON);
                                        setsegmentvisibility(currseg + PAGEOFFSE
                                        }
                                setsegmentvisibility(currseg,ON);
                                setsegmentvisibility(currseg + PAGEOFFSET,ON);
                        endbatchofupdates();
                        break;
                case '<':                               /* previous frame */
                        beginbatchofupdates();
                                setsegmentvisibility(currseg,OFF);
                                setsegmentvisibility(currseg + PAGEOFFSET,OFF);
                                --currseg;
                                if (currseg < currface)
                                        {
                                        currseg = currface;
                                        setsegmentvisibility(currseg,ON);
                                        setsegmentvisibility(currseg + PAGEOFFSE
                                        }
                                setsegmentvisibility(currseg,ON);
                                setsegmentvisibility(currseg + PAGEOFFSET,ON);
                        endbatchofupdates();
                        break;
                case '2':
                case '3':
                case '4':
                default:
                        /*      not currently implemented - therefore NOP
                         *
                         *      for cases 2,3,4 should turn current seg and page
                         *      off, and then turn on seg for first frame of tha
                         *  area.
                         *
                         *      default action should remain a NOP, or perhaps
                         *  print a warning message.
                         */
                        break;
                        }               /* end switch */

                }       while ((*option != 'Q') && (*option != 'q'));
                                /* Q or q to quit */

        }               /* end select() */
/*****************************************************************************
 *
 *  createinstr:        create text frames and accompanying page number
 *                              frames for Instrface.
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    1100.0
#define YMIN    0.0
#define YMAX    470.0

char *qframe[] = {
        "instr1.txt",
        "instr2.txt",
        "instr3.txt",
        "instr4.txt",
        "instr5.txt",
        "instr6.txt"
```

```
extern float sx, sy, tx, ty;

VOID createinstr()
    {
    char color, *line[BUFSIZ];
        FILE *fopen(), *fp[6];
        int i, offset, segname;


        for (i=0; i < 6; ++i)
                if ((fp[i] = fopen(qframe[i], "r")) == NULL)
                        error(qframe[i]);

        for (i=0; i < 6; ++i)
                {

                /* create page number segment */

                setviewport2(0.65, 1.0, 0.60, 1.0);
                setwindow(0.0, 100.0, 0.0, 100.0);

                fscanf(fp[i], "%d\n", &offset);          /* read frame offset */

                segname = INSTR + PAGEOFFSET + offset;
                createretainedsegment(segname);
                        color = 'R';
                        setcolor(color);
                        setintensity(1.0);

                        fgets(line, MAXLINE, fp[i]);    /* read page number */
                        moveabs2(50.0,80.0);
                        text(line);

                closeretainedsegment();
                setsegmentvisibility(segname,OFF);

                /* create text frame */

                setviewport2(0.3, 1.0, 0.0, 0.60);
                setwindow(XMIN, XMAX, YMIN, YMAX);

                segname = INSTR + offset;
                createretainedsegment(segname);
                        setintensity(1.0);

                        color = 'R';
                        setcolor(color);

                        moveabs2(175.0, 460.0);
                        lineabs2(900.0, 460.0);

                        moveabs2(175.0, 60.0);
                        lineabs2(900.0, 60.0);

                        moveabs2(175.0, 390.0);
                        while ((fgets(line, MAXLINE, fp[i])) != NULL)
                                {
                                text(line);
                                moverel2(0.0,-45.0);
                                }

                        fclose(fp[i]);
                closeretainedsegment();
                setsegmentvisibility(segname,OFF);
                }
```

```
    }       /* end createinstr() */
/*****************************************************************************
*
*  createouter:        create text frames and accompanying page number
*                                  frames for Outerface.
*
*****************************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    1100.0
#define YMIN    0.0
#define YMAX    470.0

char *oframe[] = {
        "outer1.txt",
        "outer2.txt",
        "outer3.txt",
        "outer4.txt",
        "outer5.txt",
        "outer6.txt",
                                };
extern float sx, sy, tx, ty;

VOID createouter()
    {
    char color, *line[BUFSIZ];
        FILE *fopen(), *fp[6];
        int i, offset, segname;


        for (i=0; i < 6; ++i)
                if ((fp[i] = fopen(oframe[i], "r")) == NULL)
                        error(oframe[i]);

        for (i=0; i < 6; ++i)
                {

                /* create page number segment */

                setviewport2(0.65, 1.0, 0.60, 1.0);
                setwindow(0.0, 100.0, 0.0, 100.0);

                fscanf(fp[i], "%d\n", &offset);          /* read frame offset */

                segname = OUTER + PAGEOFFSET + offset;
                createretainedsegment(segname);
                        color = 'R';
                        setcolor(color);
                        setintensity(1.0);

                        fgets(line, MAXLINE, fp[i]);    /* read page number */
                        moveabs2(50.0,80.0);
                        text(line);

                closeretainedsegment();
                setsegmentvisibility(segname,OFF);

                /* create text frame */

                setviewport2(0.3, 1.0, 0.0, 0.60);
                setwindow(XMIN, XMAX, YMIN, YMAX);

                        OUTER   offset;
```

```
                        setintensity(1.0);

                        color = 'R';
                        setcolor(color);

                        moveabs2(175.0, 460.0);
                        lineabs2(900.0, 460.0);

                        moveabs2(175.0, 60.0);
                        lineabs2(900.0, 60.0);

                        moveabs2(175.0, 390.0);
                        while ((fgets(line, MAXLINE, fp[i])) != NULL)
                                {
                                text(line);
                                moverel2(0.0,-45.0);
                                }

                        fclose(fp[i]);
                closeretainedsegment();
                setsegmentvisibility(segname,OFF);
                }

        }       /* end createouter() */
/*****************************************************************************
 *
 *  createinter:        create text frames and accompanying page number
 *                                  frames for Interface.
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    1100.0
#define YMIN    0.0
#define YMAX    470.0

char *itframe[] = {
        "inter1.txt",
        "inter2.txt",
        "inter3.txt",
        "inter4.txt",
        "inter5.txt",
        "inter6.txt",
                                };
extern float sx, sy, tx, ty;

VOID createinter()
    {
    char color, *line[BUFSIZ];
        FILE *fopen(), *fp[6];
        int i, offset, segname;


        for (i=0; i < 6; ++i)
                if ((fp[i] = fopen(itframe[i], "r")) == NULL)
                        error(itframe[i]);

        for (i=0; i < 6; ++i)
                {

                /* create page number segment */

                setviewport2(0.65, 1.0, 0.60, 1.0);
```

```
                setwindow(0.0, 100.0, 0.0, 100.0);

                fscanf(fp[i], "%d\n", &offset);          /* read frame offset

                segname = INTER + PAGEOFFSET + offset;
                createretainedsegment(segname);
                        cclor = 'R';
                        setcolor(color);
                        setintensity(1.0);

                        fgets(line, MAXLINE, fp[i]);     /* read page number *
                        moveabs2(50.0,20.0);
                        text(line);

                closeretainedsegment();
                setsegmentvisibility(segname,OFF);

                /* create text frame */

                setviewport2(0.3, 1.0, 0.0, 0.60);
                setwindow(XMIN, XMAX, YMIN, YMAX);

                segname = INTER + offset;
                createretainedsegment(segname);
                        setintensity(1.0);

                        color = 'R';
                        setcolor(color);

                        moveabs2(175.0, 460.0);
                        lineabs2(900.0, 460.0);

                        moveabs2(175.0, 60.0);
                        lineabs2(900.0, 60.0);

                        moveabs2(175.0, 390.0);
                        while ((fgets(line, MAXLINE, fp[i])) != NULL)
                                {
                                text(line);
                                moverel2(0.0,-45.0);
                                }

                        fclose(fp[i]);
                closeretainedsegment();
                setsegmentvisibility(segname,OFF);
                }

        }       /* end createinter() */
/*****************************************************************************
 *
 *  createinner:        create text frames and accompanying page number
 *                              frames for Innerface.
 *
 *****************************************************************************/

#include "local.h"
#include "defs.h"
#define XMIN    0.0
#define XMAX    1100.0
#define YMIN    0.0
#define YMAX    470.0

char *inframe[] = {
        "inner1.txt",
        "inner2.txt"
```

```
    "inner5.txt",
    "inner6.txt",
                        };
extern float sx, sy, tx, ty;

ID createinner()
  {
  char color, *line[BUFSIZ];
      FILE *fopen(), *fp[6];
      int i, offset, segname;


      for (i=0; i < 6; ++i)
              if ((fp[i] = fopen(inframe[i], "r")) == NULL)
                      error(inframe[i]);

      for (i=0; i < 6; ++i)
              {

              /* create page number segment */

              setviewport2(0.65, 1.0, 0.60, 1.0);
              setwindow(0.0, 100.0, 0.0, 100.0);

              fscanf(fp[i], "%d\n", &offset);          /* read frame offset */

              segname = INNER + PAGEOFFSET + offset;
              createretainedsegment(segname);
                      color = 'R';
                      setcolor(color);
                      setintensity(1.0);

                      fgets(line, MAXLINE, fp[i]);    /* read page number */
                      moveabs2(50.0,80.0);
                      text(line);

              closeretainedsegment();
              setsegmentvisibility(segname,OFF);

              /* create text frame */

              setviewport2(0.3, 1.0, 0.0, 0.60);
              setwindow(XMIN, XMAX, YMIN, YMAX);

              segname = INNER + offset;
              createretainedsegment(segname);
                      setintensity(1.0);

                      color = 'R';
                      setcolor(color);

                      moveabs2(175.0, 460.0);
                      lineabs2(900.0, 460.0);

                      moveabs2(175.0, 60.0);
                      lineabs2(900.0, 60.0);

                      moveabs2(175.0, 390.0);
                      while ((fgets(line, MAXLINE, fp[i])) != NULL)
                              {
                              text(line);
                              moverel2(0.0,-45.0);
                              }

                      fclose(fp[i]);
```

```
                closeretainedsegment();
                setsegmentvisibility(segname,OFF);
                }

        }       /* end createinner() */

/*****************************************************************************
 *
 *      transforms.c    -       transformation routines
 *
 *****************************************************************************/

#define         FLOAT   float
#define         VOID    int
typedef struct
                {
                char code;
                FLOAT x,y;
                } POINT;
VOID scale(sx,sy,pd)
        FLOAT sx,sy;    /* scale factors */
        POINT *pd;              /* pointer to data points */
        {
        pd->x *= sx;
        pd->y *= sy;
        return;
        }

VOID translate(tx,ty,pd)
        FLOAT tx,ty;    /* translate factors */
        POINT *pd;              /* pointer to data points */
        {
        pd->x += tx;
        pd->y += ty;
        return;
        }
/*****************************************************************************
 *
 *      globals.c - global variables
 *
 *****************************************************************************/

char *face[] = {
        "inner.lin",
        "inter.lin",
        "outer.lin",
        "instr.lin",
                                };

int currseg, currface, currbanner, currmenu;

/*
 *      input:  used to input point information (from either the
 *                      tablet or as redirected input from a file)
 *                      which is then stored as floating point binary
 *                      information.
 */

#include "local.h"
#define         FLOAT   float
struct point
        {
        char code;
        FLOAT x,y;
        };
```

```
        char *argv[];

        int fd;
        struct point datum, *pd;
        char s[BUFSIZ];

        if (1 < argc)
                {
                if ((fd = creat(argv[1], 0666)) <0)
                        {
                        printf("can't open %s \n", argv[1]);
                        exit(FAIL);
                        }
                }
        else
                fd = STDOUT;

        pd = &datum;
        while (getln(s, BUFSIZ) != EOF)
                {
                sscanf(s, "%c %f,%f", &pd->code, &pd->x, &pd->y);
                write(fd, pd, sizeof(struct point));
                }
        close(fd);
        exit(SUCCEEI);
}
/*
 *      input2:         used to input point information (from either the
 *                              tablet or as redirected input from a file)
 *                              which is then stored as integer binary
 *                              information.
 *
 *
 *                      In practice, "input2" was used to input the points
 *                      from the tablet, which produced integer data points.
 *                      - "adjust" was then used to create character data points
 *                      that could be edited to correct for error produced
 *                      with tablet.
 *                      After adjusting the points, "input" was then used with
 *                      its input redirected from the character file, producing
 *                      floating point data points, which all of the routines
 *                      expect.
 */

#include "local.h"
struct point
        {
        char code;
        int x,y;
        };
main(argc, argv)
        int argc;
        char *argv[];
{
        int fd;
        struct point datum, *pd;
        char s[BUFSIZ];

        if (1 < argc)
                {
                if ((fd = creat(argv[1], 0666)) <0)
                        {
                        printf("can't open %s \n", argv[1]);
                        exit(FAIL);
                        }
                }
```

```
        else
                fd = STDOUT;

        pd = &datum;
        while (getln(s, BUFSIZ) != EOF)
                {
                sscanf(s, "%c %d,%d", &pd->code, &pd->x, &pd->y);
                write(fd, pd, sizeof(struct point));
                }
        close(fd);
        exit(SUCCEED);

*
*       adjust: used to read the binary point information input
*                       as integer from the tablet, and output this to another
*                       file in character format for use with a text editor.
*/

#include "local.h"

#define VOID int    /* default return type */
typedef struct
        {
        char code;
        int x, y;
        } POINT;

VOID main(argc, argv)
        int argc;
        char *argv[];
    {
        int fd1,fd2;
        FILE *fp, *fopen();

        if (1 < argc)
                {
                if ((fd1 = open(argv[1], 0)) == -1)
                        {
                        printf("can't open %s \n", argv[1]);
                        exit(FAIL);
                        }
                if ((fd2 = open(argv[2], 0)) != -1)
                        {
                        printf("file %s already exists \n", argv[2]);
                        exit(FAIL);
                        }
                if ((fp = fopen(argv[2], "w")) == NULL)
                        {
                        printf("can't create %s \n", argv[2] );
                        exit(FAIL);
                        }
                }
        else
                {
                printf("no filenames supplied \n");
                exit(FAIL);
                }

        convert(fd1,fp);
        exit(SUCCEED);
        }

*       read binary point info from fd1, and rewrite as char in fp
*/
VOID convert(fd1,fp)
```

```
        POINT datum, *pd;

        pd = &datum;
        while ((read(fd1, pd, sizeof(POINT))) > 0)
                {
                fprintf(fp,"%c %d,%d\n", pd->code, pd->x, pd->y);
                }

        close(fd1);
        fclose(fp);
        return;
        }



#define FLOAT float
#define VOID int    /* default return type */
#define OFF             0
#define ON              1
#define INSTR   4000
#define OUTER   3000
#define INTER   2000
#define INNER   1000
#define MENU    3
#define INSTRMENU       4
#define WAIT    9
#define PAGEOFFSET 500
#define BANNEROFFSET    999
#define MAXLINE 100
#define SEGBASE 1000
/*****************************************************************************
 *
 *  some useful definitions
 *
 *****************************************************************************/

#ifndef FAIL
#include <stdio.h>
#define FAIL            1
#define FOREVER         for (;;)
#define NO              0
#define STDERR  2
#define STDIN           0
#define STDOUT  1
#define SUCCEED 0
#define YES             1
#define bool            int
#define metachar        short
#define tbool           char
#define ushort  unsigned
#define bits            ushort
#define void            int
#define getln(s, n)     ((fgets(s, n, stdin) == NULL) ? EOF : strlen(s))
#define ABS(x)  (((x) < 0) ? -(x) : (x))
#define MAX(x, y)       (((x) < (y)) ? (y) : (x))
#define MIN(x, y)       (((x) < (y)) ? (x) : (y))
#endif

# csh script to compile C programs
set I=~/lib/
cc "-I$L" -w -g $* -lm  ~graphics/core/corelib
#cc "-I$I" -w -g $* -lm  ~rtr5861/core/cgl
```